PRODUCT MANUAL

RabbitCore RCM4300

C-Programmable Analog Core Module
with *microSD™ Card* Storage and Ethernet

User's Manual

019–0163_J

RabbitCore RCM4300 User's Manual

Part Number 019-0163_J • Printed in U.S.A.

©2007–2010 Digi International Inc. • All rights reserved.

Digi International reserves the right to make changes and improvements to its products without providing notice.

Trademarks

Rabbit, RabbitCore, and Dynamic C are registered trademarks of Digi International Inc.

Rabbit 4000 is are trademark of Digi International Inc.

SD is a trademark of the SD Card Association.

The latest revision of this manual is available on the Rabbit Web site, www.rabbit.com, for free, unregistered download.

Digi International Inc.

www.rabbit.com

TABLE OF CONTENTS

Chapter 1. Introduction 6

1.1 RCM4300 Features 7
1.2 Advantages of the RCM4300 8
1.3 Development and Evaluation Tools 9
1.3.1 RCM4300 Development Kit 9
1.3.2 Software 10
1.3.3 Online Documentation 10

Chapter 2. Getting Started 11

2.1 Install Dynamic C 11
2.2 Hardware Connections 12
2.2.1 Step 1 — Prepare the Prototyping Board for Development 12
2.2.2 Step 2 — Attach Module to Prototyping Board 13
2.2.3 Step 3 — Connect Programming Cable 14
2.2.4 Step 4 — Connect Power 15
2.3 Run a Sample Program 16
2.3.1 Troubleshooting 16
2.4 Where Do I Go From Here? 17
2.4.1 Technical Support 17

Chapter 3. Running Sample Programs 18

3.1 Introduction 18
3.2 Sample Programs 19
3.2.1 Tamper Detection 20
3.2.2 Use of microSD™ Cards 20
3.2.3 Serial Communication 21
3.2.4 A/D Converter Inputs (RCM4300 only) 24
3.2.4.1 Downloading and Uploading Calibration Constants 25
3.2.5 Real-Time Clock 27
3.2.6 TCP/IP Sample Programs 27
3.2.7 FAT File System Sample Programs 27

Chapter 4. Hardware Reference 28

4.1 RCM4300 Digital Inputs and Outputs 29
4.1.1 Memory I/O Interface 35
4.1.2 Other Inputs and Outputs 35
4.2 Serial Communication 36
4.2.1 Serial Ports 36
4.2.2 Ethernet Port 37
4.2.3 Programming Port 38
4.3 Programming Cable 39
4.3.1 Changing Between Program Mode and Run Mode 39
4.3.2 Standalone Operation of the RCM4300 40

4.4 A/D Converter (RCM4300 only) 41
4.4.1 A/D Converter Power Supply 43
4.5 Other Hardware 44
4.5.1 Clock Doubler 44
4.5.2 Spectrum Spreader 44
4.6 Memory 45
4.6.1 SRAM 45
4.6.2 Flash Memory 45
4.6.3 VBAT RAM Memory 45
4.6.4 microSD™ Cards 45

Chapter 5. Software Reference 47
5.1 More About Dynamic C 47
5.2 Dynamic C Function Calls 49
5.2.1 Digital I/O 49
5.2.2 Serial Communication Drivers 49
5.2.3 Serial Flash Memory Use 50
5.2.4 User Block 52
5.2.5 SRAM Use 52
5.2.6 RCM4300 Cloning 53
5.2.7 microSD™ Card Drivers 53
5.2.8 Prototyping Board Function Calls 54
5.2.8.1 Board Initialization 54
5.2.8.2 Alerts 55
5.2.9 Analog Inputs (RCM4300 only) 56
5.3 Upgrading Dynamic C 73
5.3.1 Add-On Modules 73

Chapter 6. Using the TCP/IP Features 74
6.1 TCP/IP Connections 74
6.2 TCP/IP Primer on IP Addresses 76
6.2.1 IP Addresses Explained 78
6.2.2 How IP Addresses are Used 79
6.2.3 Dynamically Assigned Internet Addresses 80
6.3 Placing Your Device on the Network 81
6.4 Running TCP/IP Sample Programs 82
6.4.1 How to Set IP Addresses in the Sample Programs 83
6.4.2 How to Set Up your Computer for Direct Connect 84
6.5 Run the PINGME.C Sample Program 85
6.6 Running Additional Sample Programs With Direct Connect 85
6.7 Where Do I Go From Here? 86

Appendix A. RCM4300 Specifications 87
A.1 Electrical and Mechanical Characteristics 88
A.1.1 A/D Converter 92
A.1.2 Headers 93
A.2 Rabbit 4000 DC Characteristics 94
A.3 I/O Buffer Sourcing and Sinking Limit 95
A.4 Bus Loading 95
A.5 Jumper Configurations 98
A.6 Conformal Coating 100

Appendix B. Prototyping Board 101
B.1 Introduction 102
B.1.1 Prototyping Board Features 103
B.2 Mechanical Dimensions and Layout 105
B.3 Power Supply 106

B.4 Using the Prototyping Board 107
B.4.1 Adding Other Components 109
B.4.2 Measuring Current Draw 109
B.4.3 Analog Features (RCM4300 only) 110
B.4.3.1 A/D Converter Inputs 110
B.4.3.2 Thermistor Input 112
B.4.3.3 A/D Converter Calibration 112
B.4.4 Serial Communication 113
B.4.4.1 RS-232 114
B.5 Prototyping Board Jumper Configurations 115

Appendix C. Power Supply 118
C.1 Power Supplies 118
C.1.1 Battery Backup 118
C.1.2 Battery-Backup Circuit 119
C.1.3 Reset Generator 120

Index 122

Schematics 125

1. INTRODUCTION

The RCM4300 series of RabbitCore modules is one of the next generation of core modules that take advantage of new Rabbit® 4000 features such as hardware DMA, clock speeds of up to 60 MHz, I/O lines shared with up to six serial ports and four levels of alternate pin functions that include variable-phase PWM, auxiliary I/O, quadrature decoder, and input capture. Coupled with more than 500 new opcode instructions that help to reduce code size and improve processing speed, this equates to a core module that is fast, efficient, and the ideal solution for a wide range of embedded applications. The RCM4300 also features an integrated 10/100Base-T Ethernet port, an optional A/D converter, and removable ("hot-swappable") memory cards. .

Each production model has a Development Kit with the essentials that you need to design your own microprocessor-based system, and includes a complete Dynamic C software development system. The Development Kits also contains a Prototyping Board that will allow you to evaluate the specific RCM4300 module and to prototype circuits that interface to the module. You will also be able to write and test software for the RCM4300 modules.

Throughout this manual, the term RCM4300 refers to the complete series of RCM4300 RabbitCore modules unless other production models are referred to specifically.

The RCM4300 has a Rabbit 4000 microprocessor operating at up to 58.98 MHz, a fast program-execution SRAM, data SRAM, serial flash memory, an 8-channel A/D converter, two clocks (main oscillator and timekeeping), and the circuitry necessary for reset and management of battery backup of the Rabbit 4000's internal real-time clock and 512K of static RAM. One 50-pin header brings out the Rabbit 4000 I/O bus lines, parallel ports, A/D converter channels, and serial ports.

The RCM4300's mass-storage capabilities make them suited to running the optional Dynamic C FAT file system module where data are stored and handled using the same directory file structure commonly used on PCs. A removable *microSD™ Card* can be hot-

swapped to transfer data quickly and easily using a standardized file system that can be read away from the RCM4300 installation.

The RCM4300 receives its +3.3 V power from the customer-supplied motherboard on which it is mounted. The RCM4300 can interface with all kinds of CMOS-compatible digital devices through the motherboard.

1.1 RCM4300 Features

- Small size: 1.84" × 2.85" × 0.84" (47 mm × 72 mm × 21 mm)
- Microprocessor: Rabbit 4000 running at up to 58.98 MHz
- Up to 28 or 36 general-purpose I/O lines configurable with up to four alternate functions
- 3.3 V I/O lines with low-power modes down to 2 kHz
- Up to six CMOS-compatible serial ports — four ports are configurable as a clocked serial ports (SPI), and one or two ports are configurable as SDLC/HDLC serial ports.
- Combinations of up to eight single-ended or four differential 12-bit analog inputs (RCM4300 only)
- Alternate I/O bus can be configured for 8 data lines and 6 address lines (shared with parallel I/O lines), I/O read/write
- Up to 2 MB flash memory, up to 1 MB fast program-execution SRAM, and 512K data SRAM
- Removable *microSD™ Card* memory that may be used with the standardized directory structure supported by the Dynamic C FAT File System module and may be read with a standard Windows SD Card reader
- Real-time clock
- Watchdog supervisor

There are two RCM4300 production models. Table 1 summarizes their main features.

Table 1. RCM4300 Features

Feature	RCM4300	RCM4310
Microprocessor	Rabbit® 4000 at 58.98 MHz	
Data SRAM	512K	
Fast Program-Execution SRAM	1MB	512K
Serial Flash Memory (program)	2MB	1MB
Flash Memory (data storage)	*microSD™ Card* 128MB–1GB	*microSD™ Card* 128MB–1GB
A/D Converter	12 bits	—

Table 1. RCM4300 Features (continued)

Feature	RCM4300	RCM4310
Serial Ports	5 shared high-speed, CMOS-compatible ports: • all 5 configurable as asynchronous (with IrDA), 4 as clocked serial (SPI), and 1 as SDLC/HDLC • 1 clocked serial port shared with programming port • 1 clocked serial port shared with A/D converter, serial flash, and microSD™ card	6 shared high-speed, CMOS-compatible ports: • all 6 configurable as asynchronous (with IrDA), 4 as clocked serial (SPI), and 2 as SDLC/HDLC • 1 clocked serial port shared with programming port • 1 clocked serial port shared with serial flash and microSD™ card

The RCM4300 is programmed over a standard PC USB port through a programming cable supplied with the Development Kit.

NOTE: The RabbitLink cannot be used to program RabbitCore modules based on the Rabbit 4000 microprocessor.

Appendix A provides detailed specifications for the RCM4300.

1.2 Advantages of the RCM4300

- Fast time to market using a fully engineered, “ready-to-run/ready-to-program” microprocessor core.
- Competitive pricing when compared with the alternative of purchasing and assembling individual components.
- Easy C-language program development and debugging
- Rabbit Field Utility to download compiled Dynamic C .bin files, and cloning board options for rapid production loading of programs.
- Generous memory size allows large programs with tens of thousands of lines of code, and substantial data storage.
- Integrated Ethernet port for network connectivity, with royalty-free TCP/IP software.

1.3 Development and Evaluation Tools

1.3.1 RCM4300 Development Kit

The RCM4300 Development Kit contains the hardware essentials you will need to use the RCM4300 module. The items in the Development Kit and their use are as follows.

- RCM4300 module.
- Prototyping Board.
- 1 GB *microSD™ Card* with SD Card adapter.
- Universal AC adapter, 12 V DC, 1 A (includes Canada/Japan/U.S., Australia/N.Z., U.K., and European style plugs).
- USB programming cable with 10-pin header.
- Cat. 5 Ethernet crossover cable.
- 10-pin header to DB9 serial cable.
- ***Dynamic C***® CD
- ***Getting Started*** instructions.
- A bag of accessory parts for use on the Prototyping Board.
- ***Rabbit 4000 Processor Easy Reference*** poster.
- Registration card.

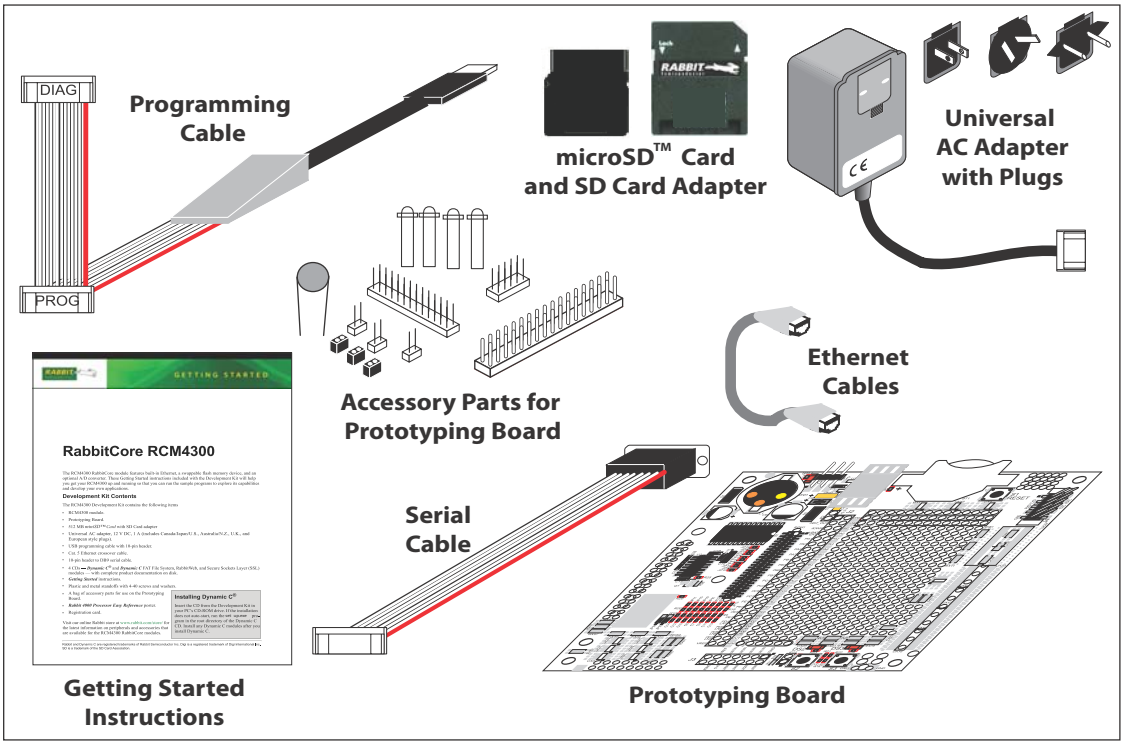


Figure 1. RCM4300 Development Kit

1.3.2 Software

The RCM4300 is programmed using version 10.21 or later of Dynamic C. A compatible version is included on the Development Kit CD-ROM.

Also now included on the CD, are the add-on Dynamic C modules containing the popular μC/OS-II real-time operating system, the FAT file system, as well as PPP, Advanced Encryption Standard (AES), and other select libraries. In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support module is also available for purchase. Visit our Web site at www.rabbit.com or contact your Rabbit Semiconductor sales representative or authorized distributor for further information.

1.3.3 Online Documentation

The online documentation is installed along with Dynamic C, and an icon for the documentation menu is placed on the workstation's desktop. Double-click this icon to reach the menu. If the icon is missing, use your browser to find and load **default.htm** in the **docs** folder, found in the Dynamic C installation folder.

The latest versions of all documents are always available for free, unregistered download from our Web sites as well.

2. GETTING STARTED

This chapter describes the RCM4300 hardware in more detail, and explains how to set up and use the accompanying Prototyping Board.

NOTE: This chapter (and this manual) assume that you have the RCM4300 Development Kit. If you purchased an RCM4300 module by itself, you will have to adapt the information in this chapter and elsewhere to your test and development setup.

2.1 Install Dynamic C

To develop and debug programs for the RCM4300 series of modules (and for all other Rabbit Semiconductor hardware), you must install and use Dynamic C.

If you have not yet installed Dynamic C version 10.21 (or a later version), do so now by inserting the Dynamic C CD from the Development Kit in your PC's CD-ROM drive. If autorun is enabled, the CD installation will begin automatically.

If autorun is disabled or the installation does not start, use the Windows **Start | Run** menu or Windows Disk Explorer to launch **`setup.exe`** from the root folder of the CD-ROM.

The installation program will guide you through the installation process. Most steps of the process are self-explanatory.

Once your installation is complete, you will have up to three new icons on your PC desktop. One icon is for Dynamic C, another opens the documentation menu, and the third is for the Rabbit Field Utility, a tool used to download precompiled software to a target system.

2.2 Hardware Connections

There are four steps to connecting the Prototyping Board for use with Dynamic C and the sample programs:

1. Prepare the Prototyping Board for Development.
2. Attach the RCM4300 module to the Prototyping Board.
3. Connect the programming cable between the RCM4300 and the PC.
4. Connect the power supply to the Prototyping Board.

2.2.1 Step 1 — Prepare the Prototyping Board for Development

Snap in four of the plastic standoffs supplied in the bag of accessory parts from the Development Kit in the holes at the corners as shown in Figure 2.

NOTE: Pay attention to use the hole that is pointed out towards the bottom left of the Prototyping Board since the hole below it is used for a standoff when mounting the RCM4300 on the Prototyping Board.

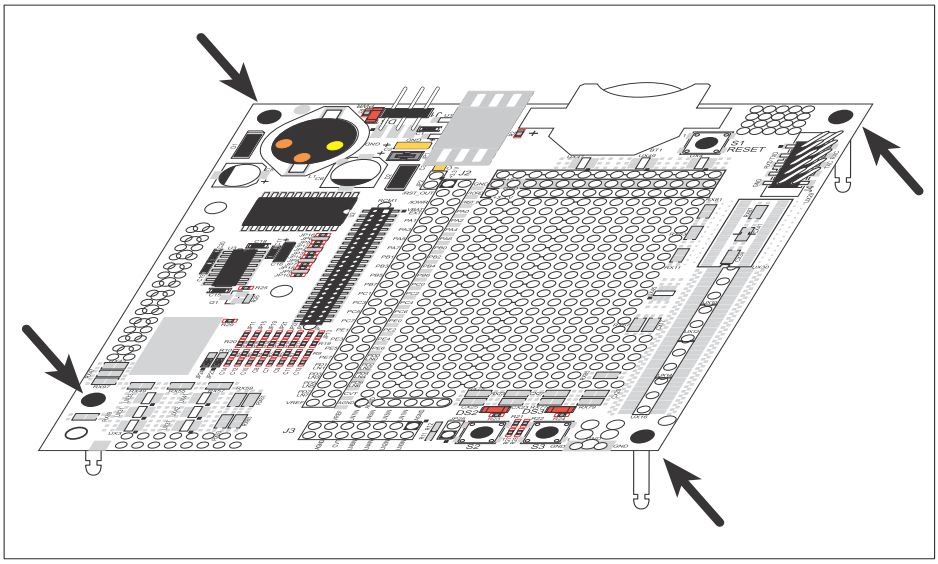


Figure 2. Insert Standoffs

2.2.2 Step 2 — Attach Module to Prototyping Board

Turn the RCM4300 module so that the mounting holes line up with the corresponding holes on the Prototyping Board. Insert the metal standoffs as shown in Figure 3, secure them from the bottom using the 4-40 × 3/16 screws and washers, then insert the module's header J4 on the bottom side into socket RCM1 on the Prototyping Board.

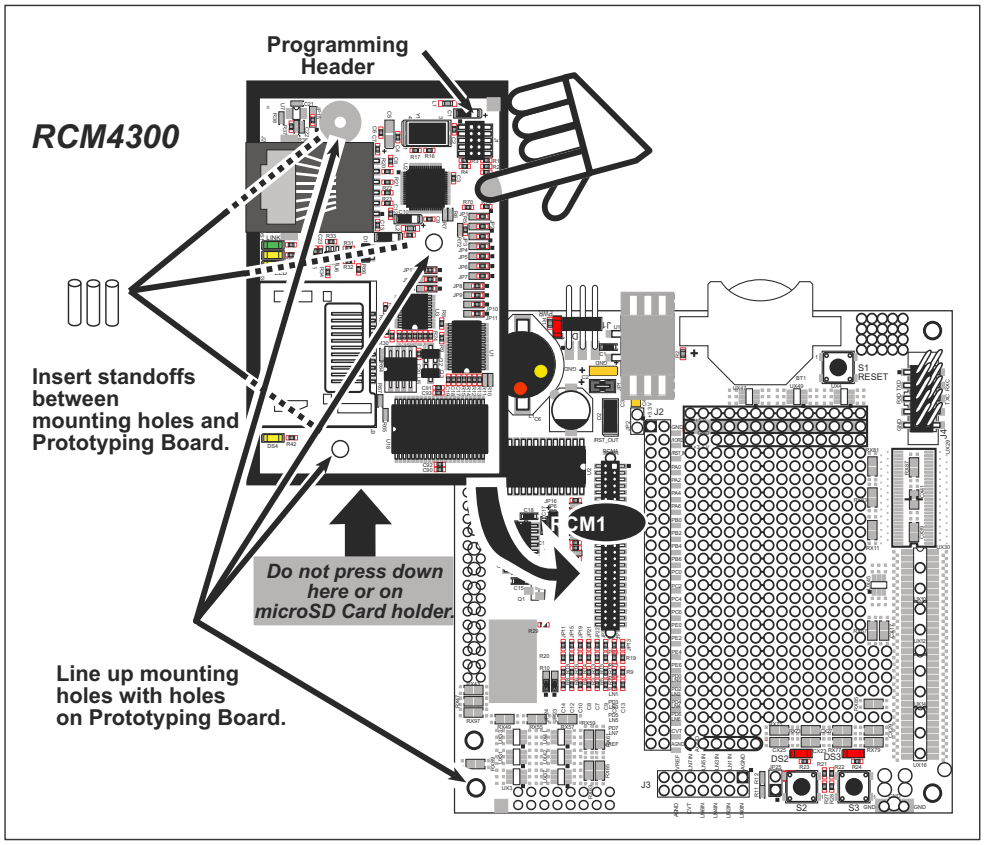

Programming Header
RCM4300
Insert standoffs between mounting holes and Prototyping Board.
Do not press down here or on microSD Card holder.
Line up mounting holes with holes on Prototyping Board.
RCM1

Figure 3. Install the Module on the Prototyping Board

NOTE: It is important that you line up the pins on header J4 of the module exactly with socket RCM1 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the module will not work. Permanent electrical damage to the module may also result if a misaligned module is powered up.

Press the module's pins gently into the Prototyping Board socket—press down in the area above the header pins. For additional integrity, you may secure the RCM4300 to the standoffs from the top using the remaining three screws and washers.

2.2.3 Step 3 — Connect Programming Cable

The programming cable connects the module to the PC running Dynamic C to download programs and to monitor the module during debugging.

Connect the 10-pin connector of the programming cable labeled **PROG** to header J1 on the RCM4300 as shown in Figure 4. Be sure to orient the marked (usually red) edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is presently not supported by the RCM4300.)

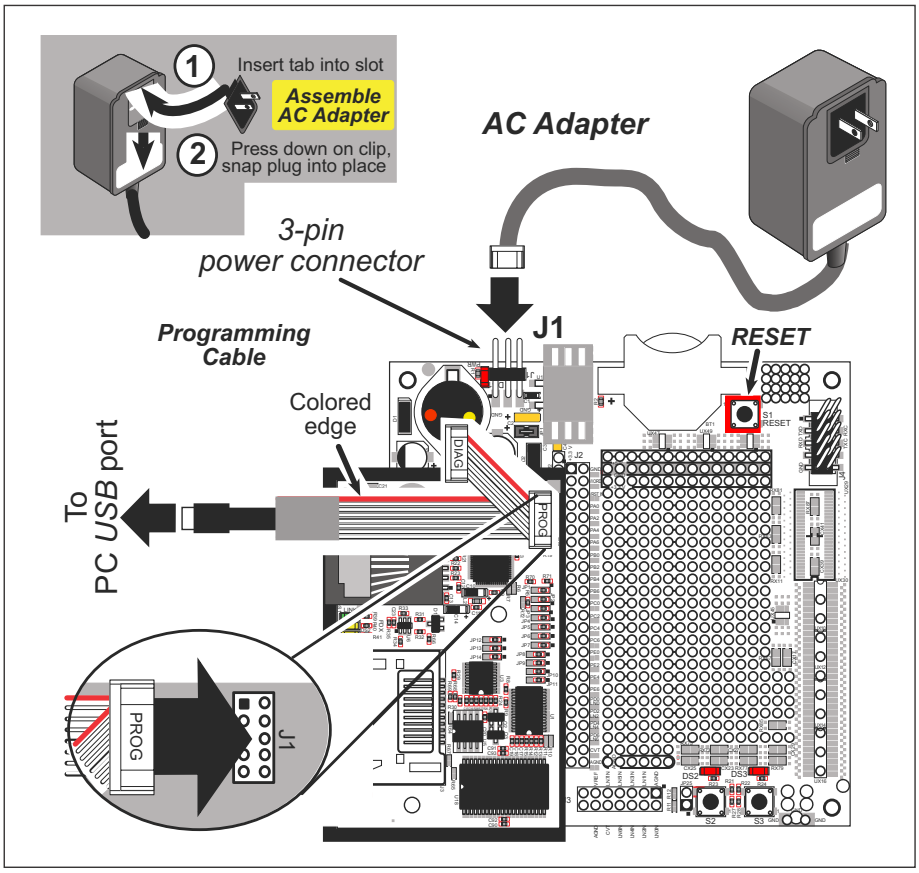


Figure 4. Connect Programming Cable and Power Supply

NOTE: Never disconnect the programming cable by pulling on the ribbon cable. Carefully pull on the connector to remove it from the header.

Connect the other end of the programming cable to an available USB port on your PC or workstation.

Your PC should recognize the new USB hardware, and the LEDs in the shrink-wrapped area of the USB programming cable will flash — if you get an error message, you will have to install USB drivers. Drivers for Windows XP are available in the Dynamic C **`Drivers\Rabbit USB Programming Cable\WinXP_2K`** folder — double-click **`DPInst.exe`** to install the USB drivers. Drivers for other operating systems are available online at www.ftdichip.com/Drivers/VCP.htm.

2.2.4 Step 4 — Connect Power

Once all the other connections have been made, you can connect power to the Prototyping Board.

First, prepare the AC adapter for the country where it will be used by selecting the plug. The RCM4300 Development Kit presently includes Canada/Japan/U.S., Australia/N.Z., U.K., and European style plugs. Snap in the top of the plug assembly into the slot at the top of the AC adapter as shown in Figure 4, then press down on the spring-loaded clip below the plug assembly to allow the plug assembly to click into place. Release the clip to secure the plug assembly in the AC adapter.

Connect the AC adapter to 3-pin header J1 on the Prototyping Board as shown in Figure 4. The connector may be attached either way as long as it is not offset to one side—the center pin of J1 is always connected to the positive terminal, and either edge pin is ground.

Plug in the AC adapter. The **PWR** LED on the Prototyping Board next to the power connector at J1 should light up. The RCM4300 and the Prototyping Board are now ready to be used.

> **NOTE:** A **RESET** button is provided on the Prototyping Board next to the battery holder to allow a hardware reset without disconnecting power.

To power down the Prototyping Board, unplug the power connector from J1. You should disconnect power before making any circuit adjustments in the prototyping area, changing any connections to the board, or removing the RCM4300 from the Prototyping Board.

2.3 Run a Sample Program

Once the RCM4300 is connected as described in the preceding pages, start Dynamic C by double-clicking on the Dynamic C icon on your desktop or in your **Start** menu. Select **Code and BIOS in RAM** on the "Compiler" tab in the Dynamic C **Options > Project Options** menu. Then click on the "Communications" tab and verify that **Use USB to Serial Converter** is selected to support the USB programming cable. Click **OK**.

> **NOTE:** The **Code and BIOS in RAM** BIOS memory compiler option is recommended while debugging for faster download times. Remember to recompile the working application using the **Code and BIOS in Flash, Run in RAM** option once you are ready to use the RCM4300 in a standalone installation.

Determine which COM port was assigned to the USB programming cable on your PC. Open **Control Panel > System > Hardware > Device Manager > Ports** and identify which COM port is used for the USB connection. In Dynamic C, select **Options > Project Options**, then select this COM port on the **Communications** tab, then click **OK**. You may type the COM port number followed by **Enter** on your computer keyboard if the COM port number is outside the range on the dropdown menu.

Use the **File** menu to open the sample program **PONG.C**, which is in the Dynamic C **SAMPLES** folder. Press function key **F9** to compile and run the program. The **STDIO** window will open on your PC and will display a small square bouncing around in a box.

This program shows that the CPU is working. The sample program described in Section 6.5, "Run the PINGME.C Sample Program," tests the TCP/IP portion of the board.

2.3.1 Troubleshooting

If you receive the message **No Rabbit Processor Detected**, the programming cable may be connected to the wrong COM port, a connection may be faulty, or the target system may not be powered up. First, check to see that the power LED on the Prototyping Board is lit. If the LED is lit, check both ends of the programming cable to ensure that it is firmly plugged into the PC and the programming header on the RCM4300 with the marked (colored) edge of the programming cable towards pin 1 of the programming header. Ensure that the module is firmly and correctly installed in its connectors on the Prototyping Board.

If Dynamic C appears to compile the BIOS successfully, but you then receive a communication error message when you compile and load a sample program, it is possible that your PC cannot handle the higher program-loading baud rate. Try changing the maximum download rate to a slower baud rate as follows.

- Locate the **Serial Options** dialog on the "Communications" tab in the Dynamic C **Options > Project Options** menu. Select a slower Max download baud rate. Click **OK** to save.

If a program compiles and loads, but then loses target communication before you can begin debugging, it is possible that your PC cannot handle the default debugging baud rate. Try lowering the debugging baud rate as follows.

- Locate the **Serial Options** dialog on the "Communications" tab in the Dynamic C **Options > Project Options** menu. Choose a lower debug baud rate. Click **OK** to save.

Press **<Ctrl-Y>** to force Dynamic C to recompile the BIOS. You should receive a **`Bios compiled successfully`** message once this step is completed successfully.

2.4 Where Do I Go From Here?

If the sample program ran fine, you are now ready to go on to the sample programs in Chapter 3 and to develop your own applications. The sample programs can be easily modified for your own use. The user's manual also provides complete hardware reference information and software function calls for the RCM4300 series of modules and the Prototyping Board.

For advanced development topics, refer to the ***Dynamic C User's Manual***, also in the online documentation set.

2.4.1 Technical Support

NOTE: If you purchased your RCM4300 through a distributor or through a Rabbit Semiconductor partner, contact the distributor or partner first for technical support.

If there are any problems at this point:

- Use the Dynamic C **Help** menu to get further assistance with Dynamic C.
- Check the Rabbit Semiconductor Technical Bulletin Board and forums at www.rabbit.com/support/bb/ and at www.rabbit.com/forums/.
- Use the Technical Support e-mail form at www.rabbit.com/support/.

3. RUNNING SAMPLE PROGRAMS

To develop and debug programs for the RCM4300 (and for all other Rabbit Semiconductor hardware), you must install and use Dynamic C. This chapter provides a tour of its major features with respect to the RCM4300.

3.1 Introduction

To help familiarize you with the RCM4300 modules, Dynamic C includes several sample programs. Loading, executing and studying these programs will give you a solid hands-on overview of the RCM4300's capabilities, as well as a quick start with Dynamic C as an application development tool.

> **NOTE:** The sample programs assume that you have at least an elementary grasp of ANSI C. If you do not, see the introductory pages of the ***Dynamic C User's Manual*** for a suggested reading list.

In order to run the sample programs discussed in this chapter and elsewhere in this manual,

1. Your module must be plugged in to the Prototyping Board as described in Chapter 2, "Getting Started."
2. Dynamic C must be installed and running on your PC.
3. The programming cable must connect the programming header on the module to your PC.
4. Power must be applied to the module through the Prototyping Board.

Refer to Chapter 2, "Getting Started," if you need further information on these steps.

To run a sample program, open it with the **File** menu (if it is not still open), then compile and run it by pressing **F9**.

Each sample program has comments that describe the purpose and function of the program. Follow the instructions at the beginning of the sample program.

Complete information on Dynamic C is provided in the ***Dynamic C User's Manual***.

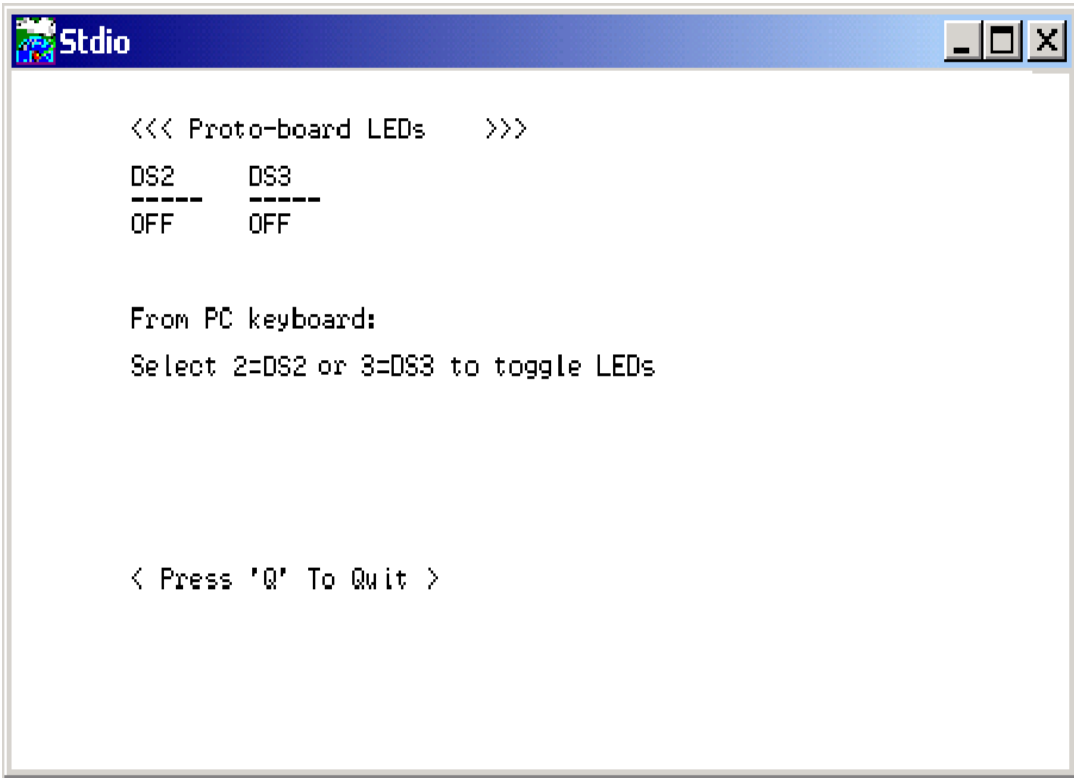
3.2 Sample Programs

Of the many sample programs included with Dynamic C, several are specific to the RCM4300 modules. These programs will be found in the **SAMPLES\RCM4300** folder.

- **CONTROLLED.C**—Demonstrates use of the digital outputs by having you turn LEDs DS2 and DS3 on the Prototyping Board on or off from the **STDIO** window on your PC.

  Parallel Port B bit 2 = LED DS2
  Parallel Port B bit 3 = LED DS3

  Once you compile and run **CONTROLLED.C**, the following display will appear in the Dynamic C **STDIO** window.



  Press "2" or "3" on your keyboard to select LED DS2 or DS3 on the Prototyping Board. Then follow the prompt in the Dynamic C **STDIO** window to turn the LED ON or OFF. A logic low will light up the LED you selected.

- **FLASHLED1.C**—demonstrates the use of assembly language to flash LEDs DS2 and DS3 on the Prototyping Board at different rates. Once you have compiled and run this program, LEDs DS2 and DS3 will flash on/off at different rates.
- **FLASHLED2.C**—demonstrates the use of cofunctions and costatements to flash LEDs DS2 and DS3 on the Prototyping Board at different rates. Once you have compiled and run this program, LEDs DS2 and DS3 will flash on/off at different rates.

- **TOGGLESWITCH.C**—demonstrates the use of costatements to detect switch presses using the press-and-release method of debouncing. LEDs DS2 and DS3 on the Prototyping Board are turned on and off when you press switches S2 and S3. S2 and S3 are controlled by PB4 and PB5 respectively.

Once you have loaded and executed these five programs and have an understanding of how Dynamic C and the RCM4300 modules interact, you can move on and try the other sample programs, or begin building your own.

3.2.1 Tamper Detection

The tamper detection feature of the Rabbit 4000 microprocessor can be used to detect any attempt to enter the bootstrap mode. When such an attempt is detected, the VBAT RAM memory on the Rabbit 4000 chip is erased. The serial bootloader on RCM4300 RabbitCore modules uses the bootstrap mode to load the SRAM, which erases the VBAT RAM memory on any reset, and so it cannot be used for tamper detection. Therefore, no tamper detection sample program is available for RCM4300 RabbitCore modules.

3.2.2 Use of *microSD*™ Cards

The following sample program can be found in the **SAMPLES\RCM4300\SD_Flash** folder.

- **SDFLASH_INSPECT.C**—This program is a utility for inspecting the contents of a *microSD*™ *Card*. It provides examples of both reading and writing pages or sectors to the *microSD*™ *Card*. When the sample program starts running, it attempts to initialize the *microSD*™ *Card* on Serial Port B. The following five commands are displayed in the Dynamic C **STDIO** window if a *microSD*™ *Card* is found:

  p — print out the contents of a specified page on the *microSD*™ *Card*

  r — print out the contents of a range of pages on the *microSD*™ *Card*

  c — clear (set to zero) all of the bytes in a specified page

  f — sets all bytes on the specified page to the given value

  t — write user-specified text to a selected page

  The sample program prints out a single line for a page if all bytes in the page are set to the same value. Otherwise it prints a hex/ASCII dump of the page.

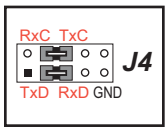
  This utility works with the *microSD*™ *Card* at its lowest level, and writing to pages will likely make the *microSD*™ *Card* unreadable by a PC. For PC compatibility, you must use the Dynamic C FAT file system module, which allows you to work with files on the *microSD*™ *Card* in a way that they will be PC-compatible.

3.2.3 Serial Communication

The following sample programs are found in the **SAMPLES\RCM4300\SERIAL** folder.

- **FLOWCONTROL.C**—This program demonstrates how to configure Serial Port C for CTS/RTS flow control with serial data coming from Serial Port D (TxD) at 115,200 bps. The serial data received are displayed in the **STDIO** window.

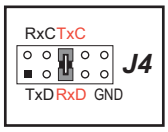
  To set up the Prototyping Board, you will need to tie TxD and RxD together on the RS-232 header at J4, and you will also tie TxC and RxC together using the jumpers supplied in the Development Kit as shown in the diagram.



  A repeating triangular pattern should print out in the **STDIO** window. The program will periodically switch flow control on or off to demonstrate the effect of flow control.

  If you have two Prototyping Boards with modules, run this sample program on the sending board, then disconnect the programming cable and reset the sending board so that the module is operating in the Run mode. Connect TxC, TxD, and GND on the sending board to RxC, RxD, and GND on the other board using jumper wires, then, with the programming cable attached to the other module, run the sample program.

- **PARITY.C**—This program demonstrates the use of parity modes by repeatedly sending byte values 0–127 from Serial Port C to Serial Port D. The program will switch between generating parity or not on Serial Port C. Serial Port D will always be checking parity, so parity errors should occur during every other sequence.

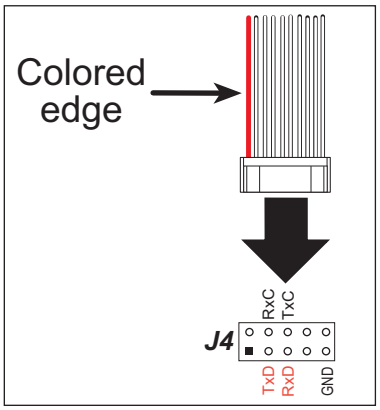


  To set up the Prototyping Board, you will need to tie TxC and RxD together on the RS-232 header at J4 using one of the jumpers supplied in the Development Kit as shown in the diagram.

  The Dynamic C **STDIO** window will display the error sequence.

- **SERDMA.C**—This program demonstrates using DMA to transfer data from a circular buffer to the serial port and vice versa. The Dynamic C **STDIO** window is used to view or clear the buffer.

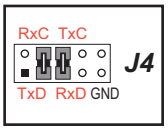
  Before you compile and run the sample program, you will need to connect the RS-232 header at J4 to your PC as shown in the diagram using the serial to DB9 cable supplied in the Development Kit.



  Once you have compiled and run the sample program, start Tera Term or another terminal emulation program to connect to the selected PC serial port at a baud rate of 115,200 bps. You can observe the output in the Dynamic C **STDIO** window as you type in Tera Term, and you can also use the Dynamic C **STDIO** window to clear the buffer.

  The Tera Term utility can be downloaded from hp.vector.co.jp/authors/VA002416/teraterm.html.

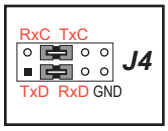
- **SIMPLE3WIRE.C**—This program demonstrates basic RS-232 serial communication. Lower case characters are sent on TxC, and are received by RxD. The received characters are converted to upper case and are sent out on TxD, are received on RxC, and are displayed in the Dynamic C **STDIO** window.



  To set up the Prototyping Board, you will need to tie TxD and RxC together on the RS-232 header at J4, and you will also tie RxD and TxC together using the jumpers supplied in the Development Kit as shown in the diagram.

- **SIMPLE5WIRE.C**—This program demonstrates 5-wire RS-232 serial communication with flow control on Serial Port C and data flow on Serial Port D.

  To set up the Prototyping Board, you will need to tie TxD and RxD together on the RS-232 header at J4, and you will also tie TxC and RxC together using the jumpers supplied in the Development Kit as shown in the diagram.

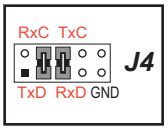


  Once you have compiled and run this program, you can test flow control by disconnecting the TxC jumper from RxC while the program is running. Characters will no longer appear in the **STDIO** window, and will display again once TxC is connected back to RxC.

  If you have two Prototyping Boards with modules, run this sample program on the sending board, then disconnect the programming cable and reset the sending board so that the module is operating in the Run mode. Connect TxC, TxD, and GND on the sending board to RxC, RxD, and GND on the other board using jumper wires, then, with the programming cable attached to the other module, run the sample program. Once you have compiled and run this program, you can test flow control by disconnecting TxC from RxC as before while the program is running. Since the J4 header locations on the two Prototyping Boards are connected with wires, there are no slip-on jumpers at J4 on either Prototyping Board.

- **SWITCHCHAR.C**—This program demonstrates transmitting and then receiving an ASCII string on Serial Ports C and D. It also displays the serial data received from both ports in the **STDIO** window.

  To set up the Prototyping Board, you will need to tie TxD and RxC together on the RS-232 header at J4, and you will also tie RxD and TxC together using the jumpers supplied in the Development Kit as shown in the diagram.



  Once you have compiled and run this program, press and release switches S2 and S3 on the Prototyping Board. The data sent between the serial ports will be displayed in the **STDIO** window.

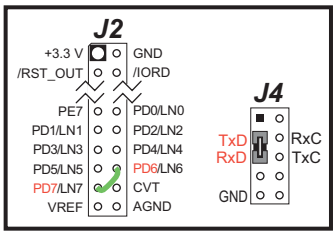
- **IOCONFIG_SWITCHECHO.C**—This program demonstrates how to set up Serial Port E, which then transmits and then receives an ASCII string when switch S2 is pressed. The echoed serial data are displayed in the Dynamic C **STDIO** window.

  Note that the I/O lines that carry the Serial Port E signals are not the Rabbit 4000 defaults. The Serial Port E I/O lines are configured by calling the library function **serEconfig()** that was generated by the Rabbit 4000 **IOCONFIG.EXE** utility program. Serial Port E is configured to use Parallel Port E bits PD6 and PD7. These signals are available on the Prototyping Board's Module Extension Header (header J2).

  Serial Port D is left in its default configuration, using Parallel Port C bits PC0 and PC1. These signals are available on the Prototyping Board's RS-232 connector (header J4). Serial Port D transmits and then receives an ASCII string when switch S3 is pressed.

  Also note that there is one library generated by **IOCONFIG.EXE** in the Dynamic C **SAMPLES\RCM4300\SERIAL** folder for the 29 MHz RCM4210.

  To set up the Prototyping Board, you will need to tie TxD and RxD together on the RS-232 header at J4 using the jumpers supplied in the Development Kit; you will also tie TxE (PD6) and RxE (PD7) together with a soldered wire or with a wire jumper if you have soldered in the IDC header supplied with the accessory parts in the Development Kit.



  Once you have compiled and run this program, press and release switches S2 or S3 on the Prototyping Board. The data echoed between the serial ports will be displayed in the **STDIO** window.

3.2.4 A/D Converter Inputs (RCM4300 only)

The following sample programs are found in the **SAMPLES\RCM4300\ADC** folder.

- **AD_CAL_ALL.C**—Demonstrates how to recalibrate all the single-ended analog input channels with one gain using two known voltages to generate the calibration constants for each channel. The constants will be written into the user block data area.

  Connect a positive voltage from 0–20 V DC (for example, the power supply positive output) to analog input channels LN0IN–LN6IN on the Prototyping Board, and connect the ground to GND. Use a voltmeter to measure the voltage, and follow the instructions in the Dynamic C **STDIO** window once you compile and run this sample program. Remember that analog input LN7 on the Prototyping Board is used with the thermistor and is not be used with this sample program.

  NOTE: The above sample program will overwrite the existing calibration constants.

- **AD_CAL_CHAN.C**—Demonstrates how to recalibrate one single-ended analog input channel with one gain using two known voltages to generate the calibration constants for that channel. The constants will be rewritten into the user block data area.

  Connect a positive voltage from 0–20 V DC (for example, the power supply positive output) to an analog input channel on the Prototyping Board, and connect the ground to GND. Use a voltmeter to measure the voltage, and follow the instructions in the Dynamic C **STDIO** window once you compile and run this sample program. Remember that analog input LN7 on the Prototyping Board is used with the thermistor and is not be used with this sample program.

  NOTE: The above sample program will overwrite the existing calibration constants for the selected channel.

- **AD_RDVOLT_ALL.C**—Demonstrates how to read all single-ended A/D input channels using previously defined calibration constants. The constants used to compute equivalent voltages are read from the user block data area, so the sample program cannot be run using the "Code and BIOS in RAM" compiler option.

  Compile and run this sample program once you have connected a positive voltage from 0–20 V DC (for example, the power supply positive output) to analog input channels LN0IN–LN6IN on the Prototyping Board, and ground to GND. Follow the prompts in the Dynamic C **STDIO** window. Raw data and the computed equivalent voltages will be displayed. Remember that analog input LN7 on the Prototyping Board is used with the thermistor and is not be used with this sample program.

- **AD_SAMPLE.C**—Demonstrates how to how to use a low level driver on single-ended inputs. The program will continuously display the voltage (averaged over 10 samples) that is present on an A/D converter channel (except LN7, which is reserved for use with a thermistor—see Appendix B.4.3.2). The constants used to compute equivalent voltages are read from the user block data area, so the sample program cannot be run using the "Code and BIOS in RAM" compiler option.

  Compile and run this sample program once you have connected a positive voltage from 0–20 V DC to an analog input (except LN7) on the Prototyping Board, and ground to GND. Follow the prompts in the Dynamic C **STDIO** window. Raw data and the computed equivalent voltages will be displayed. If you attach a voltmeter between the analog input and ground, you will be able to observe that the voltage in the Dynamic C **STDIO** window tracks the voltage applied to the analog input as you vary it.

- **THERMISTOR.C**—Demonstrates how to use analog input LN7 to calculate temperature for display to the Dynamic C **STDIO** window. This sample program assumes that the thermistor is the one included in the Development Kit whose values for beta, series resistance, and resistance at standard temperature are given in the part specification.

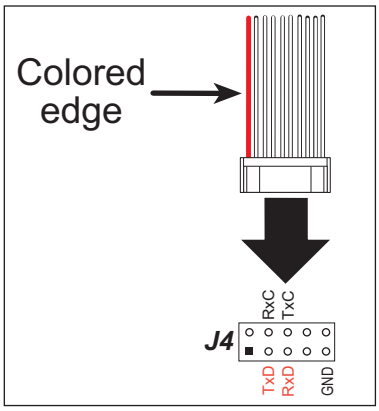
  Install the thermistor at location JP25 on the Prototyping Board before running this sample program. Observe the temperature changes shown in the Dynamic C **STDIO** window as you apply heat or cold air to the thermistor.

3.2.4.1 Downloading and Uploading Calibration Constants

The Tera Term utility called for in these sample programs can be downloaded from hp.vector.co.jp/authors/VA002416/teraterm.html.

These sample programs must be compiled to run from the fast SRAM. To do so, select **Options > Project Options** in Dynamic C, then select the "Compiler" tab, and select "**Code and BIOS in Flash, Run in RAM**" for the **BIOS Memory Setting**.

Before you compile and run these sample programs, you will also need to connect the RS-232 header at J4 to your PC as shown in the diagram using the serial to DB9 cable supplied in the Development Kit.



- **DNLOADCALIB.C**—Demonstrates how to retrieve analog calibration data to rewrite it back to the user block using a terminal emulation utility such as Tera Term.

  Start Tera Term or another terminal emulation program on your PC, and configure the serial parameters as follows.

  - Baud rate 19,200 bps, 8 bits, no parity, 1 stop bit
  - Enable **Local Echo** option
  - Feed options — **Receive = CR**, **Transmit = CR + LF**

  Now compile and run this sample program. Verify that the message "Waiting, Please Send Data file" message is being displayed in the Tera Term display window before proceeding.

  Within Tera Term, select **File-->Send File-->Path and filename**, then select the **OPEN** option within the dialog box. Once the data file has been downloaded, Tera Term will indicate whether the calibration data were written successfully.

- **UPLOADCALIB.C**—Demonstrates how to read the analog calibration constants from the user block using a terminal emulation utility such as Tera Term.

  Start Tera Term or another terminal emulation program on your PC, and configure the serial parameters as follows.

- Baud rate 19,200 bps, 8 bits, no parity, 1 stop bit
- Enable **Local Echo** option
- Feed options — **Receive = CR**, **Transmit = CR + LF**

Follow the remaining steps carefully in Tera Term to avoid overwriting previously saved calibration data when using same the file name.

- Enable the **File APPEND** option at the bottom of the dialog box
- Select the **OPEN** option at the right-hand side of the dialog box

Tera Term is now ready to log all data received on the serial port to the file you specified.

You are now ready to compile and run this sample program. A message will be displayed in the Tera Term display window once the sample program is running.

Enter the serial number you assigned to your RabbitCore module in the Tera Term display window, then press the **ENTER** key. The Tera Term display window will now display the calibration data.

Now select **CLOSE** from within the Tera Term LOG window, which will likely be a separate pop-up window minimized at the bottom of your PC screen. This finishes the logging and closes the file.

Open your data file and verify that the calibration data have been written properly. A sample is shown below.

```
Serial port transmission
========================
Uploading calibration table . . .
Enter the serial number of your controller = 9MN234
SN9MN234
ADSE
0
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
1
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
          |
          |
ADDF
0
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
2
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
          |
          |
ADMA
3
float_gain,float_offset,
4
float_gain,float_offset,
          |
          |
END
```

3.2.5 Real-Time Clock

If you plan to use the real-time clock functionality in your application, you will need to set the real-time clock. Set the real-time clock using the **SETRTCKB.C** sample program from the Dynamic C **SAMPLES\RTCLOCK** folder, using the onscreen prompts. The **RTC_TEST.C** sample program in the Dynamic C **SAMPLES\RTCLOCK** folder provides additional examples of how to read and set the real-time clock.

3.2.6 TCP/IP Sample Programs

Section 6.4 describes the TCP/IP sample programs available for the RCM4300 and how to run them.

3.2.7 FAT File System Sample Programs

The Dynamic C **SAMPLES\FileSystem\FAT** folder provides sample programs to illustrate the use of the Dynamic C implementation of the FAT file system.The FAT File System document on the Dynamic C CD describes the FAT file system in detail, describes the function calls, and discusses how to run the sample programs.

4. HARDWARE REFERENCE

Chapter 4 describes the hardware components and principal hardware subsystems of the RCM4300. Appendix A, "RCM4300 Specifications," provides complete physical and electrical specifications.

Figure 5 shows the Rabbit-based subsystems designed into the RCM4300.

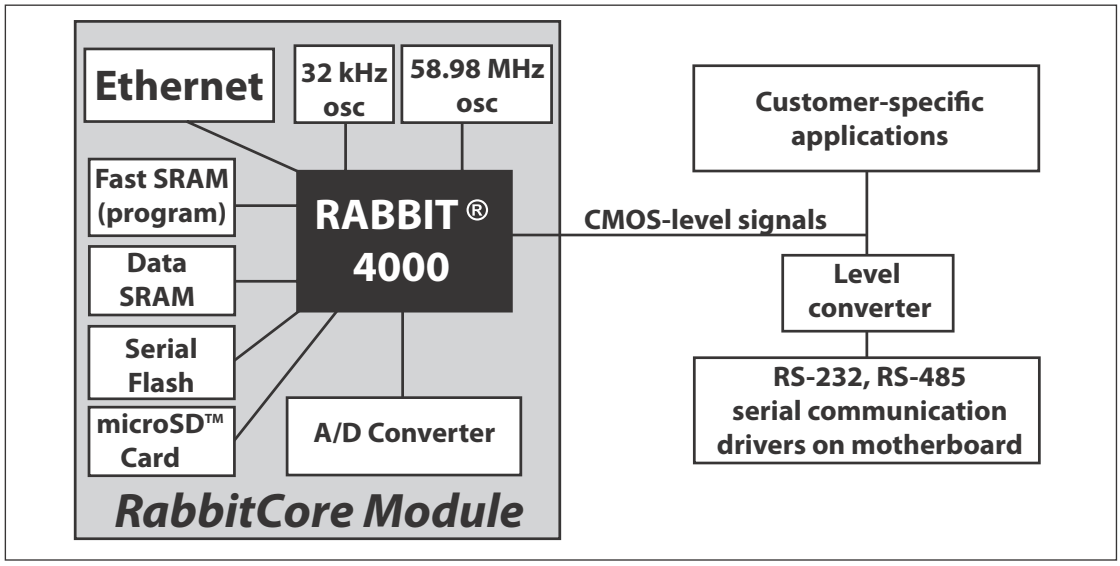


Figure 5. RCM4300 Subsystems

4.1 RCM4300 Digital Inputs and Outputs

Figure 6 shows the RCM4300 pinouts for header J4.

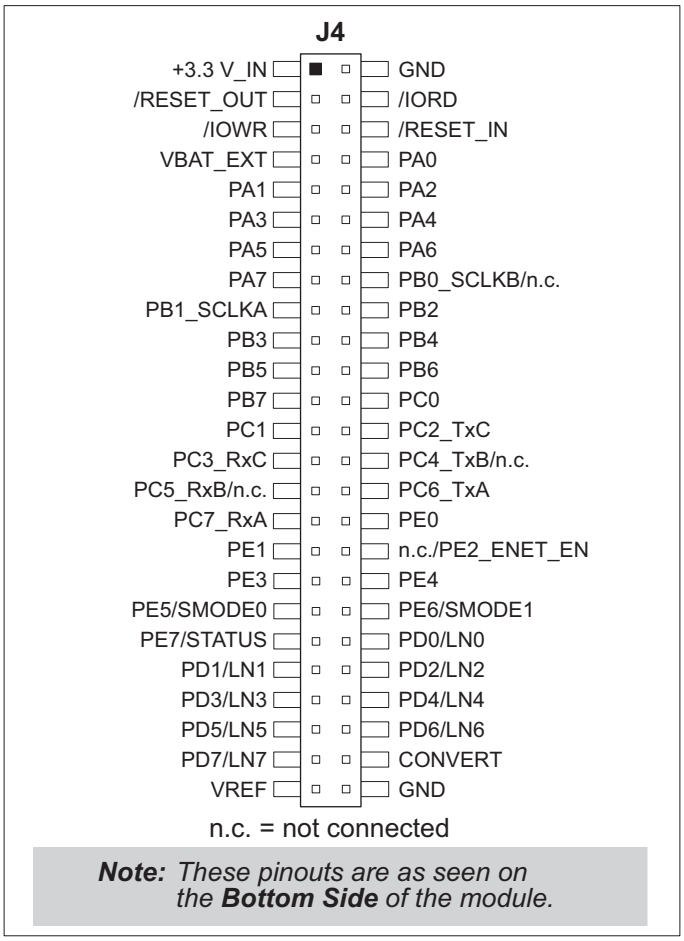


Figure 6. RCM4300 Pinout

Headers J4 is a standard 2 × 25 IDC header with a nominal 1.27 mm pitch.

Figure 7 shows the use of the Rabbit 4000 microprocessor ports in the RCM4300 modules.

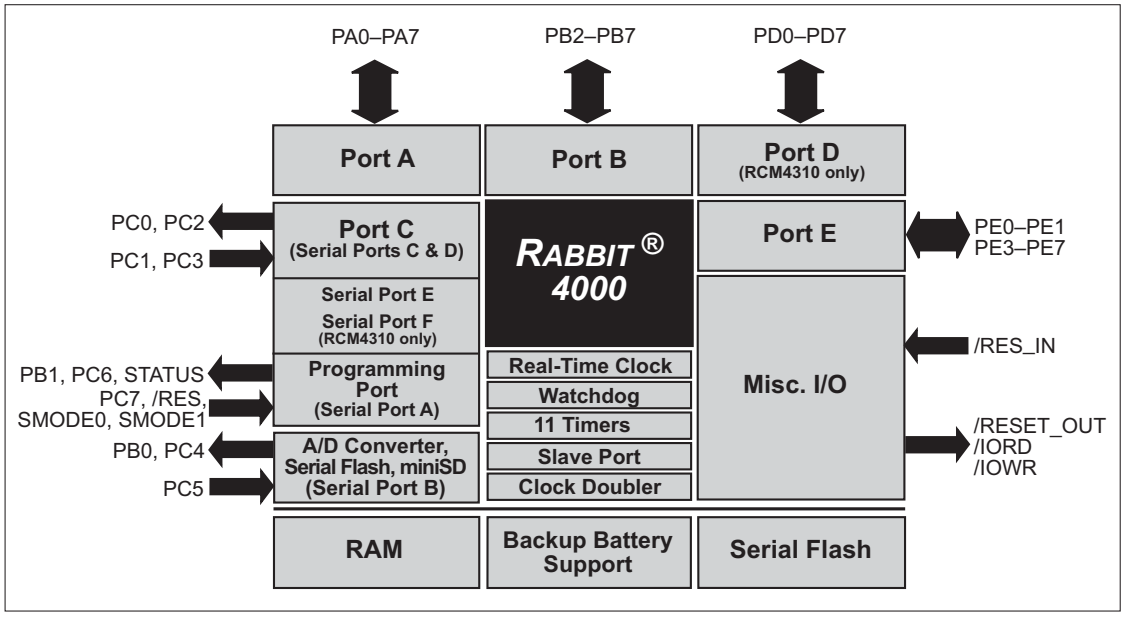


Figure 7. Use of Rabbit 4000 Ports

The ports on the Rabbit 4000 microprocessor used in the RCM4300 are configurable, and so the factory defaults can be reconfigured. Table 2 lists the Rabbit 4000 factory defaults and the alternate configurations.

Table 2. RCM4300 Pinout Configurations

Pin	Pin Name	Default Use	Alternate Use	Notes
1	+3.3 V_IN			
2	GND			
3	/RES_OUT	Reset output	Reset input	Reset output from Reset Generator or external reset input
4	/IORD	Output		External I/O read strobe
5	/IOWR	Output		External I/O write strobe
6	/RESET_IN	Input		Input to Reset Generator
7	VBAT_EXT	Battery input		
8–15	PA[0:7]	Input/Output	Slave port data bus (SD0–SD7) External I/O data bus (ID0–ID7)	
16	PB0	Input/Output	SCLKB External I/O Address IA6	SCLKB (used by serial flash and by RCM4300 A/D converter)
17	PB1	Input/Output	SCLKA External I/O Address IA7	Programming port CLKA
18	PB2	Input/Output	/SWR External I/O Address IA0	
19	PB3	Input/Output	/SRD External I/O Address IA1	
20	PB4	Input/Output	SA0 External I/O Address IA2	
21	PB5	Input/Output	SA1 External I/O Address IA3	
22	PB6	Input/Output	/SCS External I/O Address IA4	
23	PB7	Input/Output	/SLAVATN External I/O Address IA5	

Table 2. RCM4300 Pinout Configurations (continued)

Pin	Pin Name	Default Use	Alternate Use	Notes
24	PC0	Input/Output	TXD I/O Strobe I0 Timer C0 TCLKF	Serial Port D
25	PC1	Input/Output	RXD/TXD I/O Strobe I1 Timer C1 RCLKF Input Capture	
26	PC2	Input/Output	TXC/TXF I/O Strobe I2 Timer C2	Serial Port C
27	PC3	Input/Output	RXC/TXC/RXF I/O Strobe I3 Timer C3 SCLKD Input Capture	
28	PC4	Input/Output	TXB I/O Strobe I4 PWM0 TCLKE	Serial Port B (shared by serial flash and by RCM4300 A/D converter)
29	PC5	Input/Output	RXB/TXB I/O Strobe I5 PWM1 RCLKE Input Capture	
30	PC6	Input/Output	TXA/TXE I/O Strobe I6 PWM2	Programming port
31	PC7	Input/Output	RXA/TXA/RXE I/O Strobe I7 PWM3 SCLKC Input Capture	
32	PE0	Input/Output	I/O Strobe I0 A20 Timer C0 TCLKF INT0 QRD1B	

Table 2. RCM4300 Pinout Configurations (continued)

Pin	Pin Name	Default Use	Alternate Use	Notes
33	PE1	Input/Output	I/O Strobe I1 A21 Timer C1 RXD/RCLKF INT1 QRD1A Input Capture	
34	/PE2_ENET_EN	Input		Ethernet enable (not connected)
35	PE3	Input/Output	I/O Strobe I3 A23 Timer C3 RXC/RXF/SCLKD DREQ1 QRD2A Input Capture	
36	PE4	Input/Output	I/O Strobe I4 /A0 INT0 PWM0 TCLKE	
37	PE5/SMODE0	Input/Output	I/O Strobe I5 INT1 PWM1 RXB/RCLKE Input Capture	PE5 is the default configuration
38	PE6/SMODE1	Input/Output	I/O Strobe I6 PWM2 TXE DREQ0	PE6 is the default configuration
39	PE7/STATUS	Input/Output	I/O Strobe I7 PWM3 RXA/RXE/SCLKC DREQ1 Input Capture	PE7 (STATUS) is the default configuration
40–47	LN[0:7]	Analog Input		A/D converter (RCM4300 only)

Table 2. RCM4300 Pinout Configurations (continued)

Pin	Pin Name	Default Use	Alternate Use	Notes
40	PD0	Input/Output	I/O Strobe I0 Timer C0 D8 INT0 SCLKD/TCLKF QRD1B	
41	PD1	Input/Output	IA6 I/O Strobe I1 Timer C1 D9 INT1 RXD/RCLKF QRD1A Input Capture	RCM4310 only
42	PD2	Input/Output	I/O Strobe I2 Timer C2 D10 DREQ0 TXF/SCLKC QRD2B	SCLKC/Serial Port F (RCM4310 only)
43	PD3	Input/Output	IA7 I/O Strobe I3 Timer C3 D11 DREQ1 RXC/RXF QRD2A Input Capture	Serial Port F (RCM4310 only)
44	PD4	Input/Output	I/O Strobe I4 D12 PWM0 TXB/TCLKE	
45	PD5	Input/Output	IA6 I/O Strobe I5 D13 PWM1 RXB/RCLKE Input Capture	RCM4310 only

Table 2. RCM4300 Pinout Configurations (continued)

Pin	Pin Name	Default Use	Alternate Use	Notes
46	PD6	Input/Output	I/O Strobe I6 D14 PWM2 TXA/TXE	Serial Port E (RCM4310 only)
47	PD7	Input/Output	IA7 I/O Strobe I7 D15 PWM3 RXA/RXE Input Capture	
48	CONVERT	Digital Input		A/D converter (RCM4300 only)
49	VREF	Analog reference voltage		1.15 V/2.048 V/2.500 V on-chip ref. voltage (RCM4300 only)
50	GND	Ground		Analog ground

4.1.1 Memory I/O Interface

The Rabbit 4000 address lines (A0–A19) and all the data lines (D0–D7) are routed internally to the onboard SRAM chips. I/0 write (/IOWR) and I/0 read (/IORD) are available for interfacing to external devices, and are also used by the RCM4300.

Parallel Port A can also be used as an external I/O data bus to isolate external I/O from the main data bus. Parallel Port B pins PB2–PB7 can also be used as an auxiliary address bus.

When using the auxiliary I/O bus for any reason, you must add the following line at the beginning of your program.

```
#define PORTA_AUX_IO    // required to enable auxiliary I/O bus
```

Selected pins on Parallel Ports C, D, and E as specified in Table 2 may be used for input capture, quadrature decoder, DMA, and pulse-width modulator purposes.

4.1.2 Other Inputs and Outputs

The PE5–PE7 pins can be brought out to header J4 instead of the STATUS and the two SMODE pins, SMODE0 and SMODE1, as explained in Appendix A.5.

/RESET_IN is normally associated with the programming port, but may be used as an external input to reset the Rabbit 4000 microprocessor and the RCM4300 memory. /RESET_OUT is an output from the reset circuitry that can be used to reset other peripheral devices.

4.2 Serial Communication

The RCM4300 module does not have any serial driver or receiver chips directly on the board. However, a serial interface may be incorporated on the board the RCM4300 is mounted on. For example, the Prototyping Board has an RS-232 transceiver chip.

4.2.1 Serial Ports

There are six serial ports designated as Serial Ports A, B, C, D, E, and F. All six serial ports can operate in an asynchronous mode up to the baud rate of the system clock divided by 8. An asynchronous port can handle 7 or 8 data bits. A 9th bit address scheme, where an additional bit is sent to mark the first byte of a message, is also supported.

Serial Port A is normally used as a programming port, but may be used either as an asynchronous or as a clocked serial port once application development has been completed and the RCM4300 is operating in the Run Mode. Digi recommends that you limit the use of Serial Port A to a programming port to avoid any possible conflicts with the PIC microcontroller during boot-up while the program is loaded. If you intend to use Serial Port A as a regular serial port with the RCM4300 operating in the Run Mode, Serial Port A must be disconnected from the external device during boot-up in order for the program to load.

Serial Port B, shared by the RCM4300 module's serial flash, microSD™ card, and A/D converter, is set up as a clocked serial port. Since this serial port is set up for synchronous serial communication, you will lose the peripheral functionality if you try to use the serial port in the asynchronous mode.

> **NOTE:** Since Serial Port B is shared, exercise care if you attempt to use Serial Port B for other serial communication. Your application will have to manage the sharing negotiations to avoid conflicts when reading or writing to the devices already using Serial Port B. Any conflict with Serial Port B while the RCM4300 is powering up may prevent an application from loading from the serial flash when the RCM4300 powers up or resets. Do not drive or load the Serial Port B or SCLKB (PC4, PC5, and PB0) pins while the RCM4300 is powering up.

Serial Ports C and D may be used synchronously or asynchronously.

The IrDA protocol is supported by all six serial ports.

Serial Ports E and F can also be configured as SDLC/HDLC serial ports. Serial Ports E and F must be configured before they can be used. The sample program **IOCONFIG_SWITCHECHO.C** in the Dynamic C **SAMPLES\RCM4300\SERIAL** folder shows how to configure Serial Port F.

Table 3 summarizes the possible parallel port pins for the serial ports and their clocks. Remember that the Parallel Port D pins are available only on the RCM4310.

Table 3. Rabbit 4000 Serial Port and Clock Pins

Serial Port A	TXA	PC6, PC7, PD6	Serial Port E	TXE	PD6, PE6, PC6
	RXA	PC7, PD7, PE7		RXE	PD7, PE7, PC7
	SCLKA	PB1		RCLKE	PD5, PE5, PC5
Serial Port B	TXB	PC4, PC5, PD4		TCLKE	PD4, PE4, PC4
	RXB	PC5, PD5, PE5	Serial Port F	TXF	PD2, PE2, PC2
	SCLKB	PB0		RXF	PD3, PE3, PC3
Serial Port C	TXC	PC2, PC3		RCLKF	PD1, PE1, PC1
	RXC	PC3, PD3, PE3		TCLKF	PD0, PE0, PC0
	SCLKC	PD2, PE2, PE7, PC7			
Serial Port D	TXD	PC0, PC1	RCLKE and RCLKF must be selected to be on the same parallel port as TXE and TXF respectively.		
	RXD	PC1, PD1, PE1			
	SCLKD	PD0, PE0, PE3, PC3			

4.2.2 Ethernet Port

Figure 8 shows the pinout for the RJ-45 Ethernet port (J2). Note that some Ethernet connectors are numbered in reverse to the order used here.

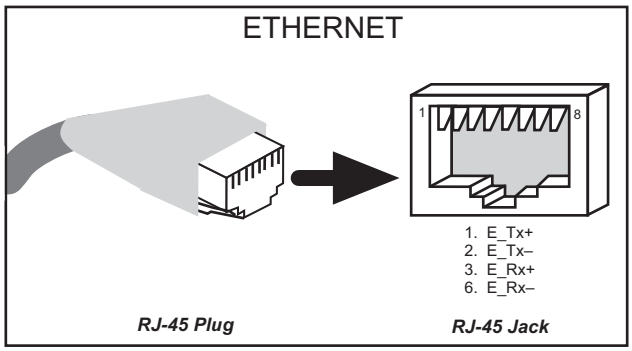


Figure 8. RJ-45 Ethernet Port Pinout

Three LEDs are placed next to the RJ-45 Ethernet jack, one to indicate Ethernet link/activity (**LINK/ACT**), one to indicate when the RCM4300 is connected to a functioning 100Base-T network (**SPEED**), and one (**FDX/COL**) to indicate that the current connection is in full-duplex mode (steady on) or that a half-duplex connection is experiencing collisions (blinks).

The RJ-45 connector is shielded to minimize EMI effects to/from the Ethernet signals.

4.2.3 Programming Port

The RCM4300 is programmed via the 10-pin header labeled J1. The programming port uses the Rabbit 4000's Serial Port A for communication. Dynamic C uses the programming port to download and debug programs.

Serial Port A is also used for the following operations.

- Cold-boot the Rabbit 4000 on the RCM4300 after a reset.
- Fast copy designated portions of flash memory from one Rabbit-based board (the master) to another (the slave) using the Rabbit Cloning Board.

In addition to Serial Port A, the Rabbit 4000 startup-mode (SMODE0, SMODE1), STATUS, and reset pins are available on the programming port.

The two startup-mode pins determine what happens after a reset—the Rabbit 4000 powers up in the asynchronous serial bootstrap/triplet mode with the programming cable attached, and in the clocked serial bootstrap/triplet mode without it. In the Run Mode, where the programming cable is not attached, the PIC microcontroller at U6 loads an initial loader that is stored on its internal flash to the RCM4300 SRAM. The PIC microcontroller communicates with the Rabbit 4000 in the clocked serial bootstrap/triplet mode to load this loader, then tells the Rabbit 4000 to exit the bootstrap mode and run the loader in SRAM, whereupon the loader loads the BIOS from the serial flash to the RCM4300 SRAM and runs it. The BIOS then loads the rest of the program from the serial flash and runs it.

The status pin is used by Dynamic C to determine whether a Rabbit microprocessor is present. The status output has three different programmable functions:

1. It can be driven low on the first op code fetch cycle.
2. It can be driven low during an interrupt acknowledge cycle.
3. It can also serve as a general-purpose output once a program has been downloaded and is running.

The reset pin is an external input that is used to reset the Rabbit 4000.

Alternate Uses of the Programming Port

All three Serial Port A signals are available as

- a synchronous serial port
- an asynchronous serial port, with the clock line usable as a general CMOS I/O pin

The programming port may also be used as a serial port once the application is running. The SMODE pins may then be used as inputs and the status pin may be used as an output.

Once the application code is running, the application must then manage sharing Serial Port A with the programming port (for example, disconnecting the external serial device during programming and debugging). The SMODE and STATUS pins are similarly not available during boot-up in order to allow the program to load.

Refer to the ***Rabbit 4000 Microprocessor User's Manual*** for more information.

4.3 Programming Cable

The programming cable is used to connect the programming port of the RCM4300 to a PC serial COM port. The programming cable converts the RS-232 voltage levels used by the PC serial port to the CMOS voltage levels used by the Rabbit 4000.

When the **PROG** connector on the programming cable is connected to the programming port on the RCM4300, programs can be downloaded and debugged over the serial interface.

4.3.1 Changing Between Program Mode and Run Mode

The RCM4300 is automatically in Program Mode when the **PROG** connector on the programming cable is attached, and is automatically in Run Mode when no programming cable is attached. When the Rabbit 4000 is reset, the operating mode is determined by the state of the SMODE pins. When the programming cable's **PROG** connector is attached, both the SMODE pins are pulled high, placing the Rabbit 4000 in the Program Mode. When the programming cable's **PROG** connector is not attached, the SMODE0 pin is pulled low and the SMODE1 pin is high so that the Rabbit 4000 powers up in the clocked serial bootstrap mode to load the program from the serial flash when the RCM4300 is operating in the Run Mode.

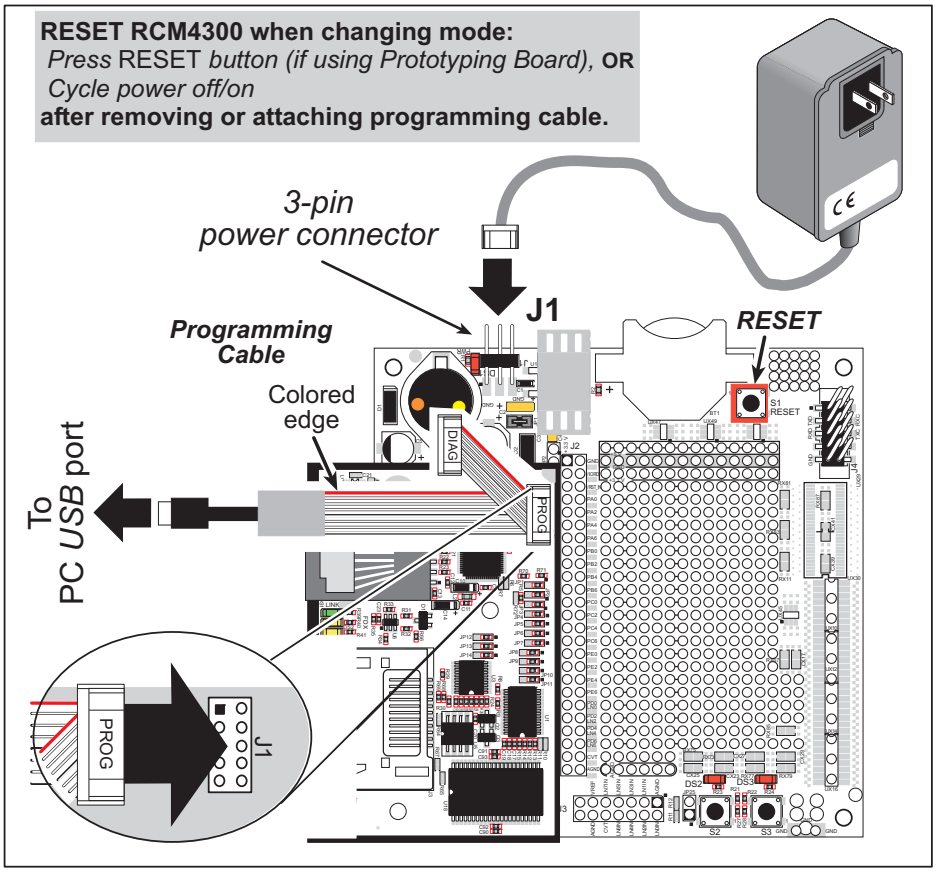


Figure 9. Switching Between Program Mode and Run Mode

A program “runs” in either mode, but can only be downloaded and debugged when the RCM4300 is in the Program Mode.

Refer to the ***Rabbit 4000 Microprocessor User’s Manual*** for more information on the programming port.

4.3.2 Standalone Operation of the RCM4300

Once the RCM4300 has been programmed successfully, remove the programming cable from the programming connector and reset the RCM4300. The RCM4300 may be reset by cycling, the power off/on or by pressing the **RESET** button on the Prototyping Board. The RCM4300 module may now be removed from the Prototyping Board for end-use installation.

CAUTION: Power to the Prototyping Board or other boards should be disconnected when removing or installing your RCM4300 module to protect against inadvertent shorts across the pins or damage to the RCM4300 if the pins are not plugged in correctly. Do not reapply power until you have verified that the RCM4300 module is plugged in correctly.

NOTE: If you used the **Code and BIOS in RAM** BIOS memory compiler option while debugging, remember to recompile the working application using the **Code and BIOS in Flash, Run in RAM** option once you are ready to use the RCM4300 in a standalone installation.

4.4 A/D Converter (RCM4300 only)

The RCM4300 has an onboard ADS7870 A/D converter whose scaling and filtering are done via the motherboard on which the RCM4300 module is mounted. The A/D converter multiplexes converted signals from eight single-ended or four differential inputs to Serial Port B on the Rabbit 4000.

The eight analog input pins, LN0–LN7, each have an input impedance of 6–7 MΩ, depending on whether they are used as single-ended or differential inputs. The input signal can range from -2 V to +2 V (differential mode) or from 0 V to +2 V (single-ended mode).

Use a resistor divider such as the one shown in Figure 10 to measure voltages above 2 V on the analog inputs.

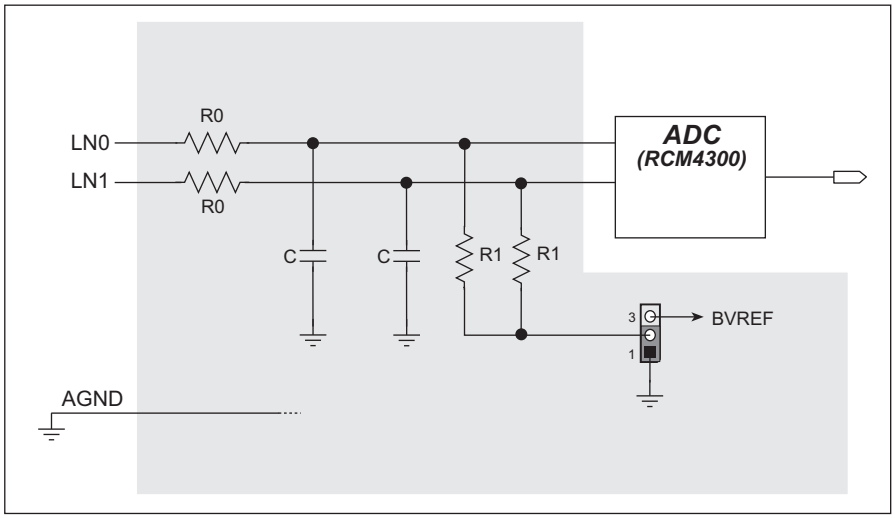


Figure 10. Resistor Divider Network for Analog Inputs

The R1 resistors are typically 20 kΩ to 100 kΩ, with a lower resistance leading to more accuracy, but at the expense of a higher current draw. The R0 resistors would then be 180 kΩ to 900 kΩ for a 10:1 attenuator. The capacitor filters noise pulses on the A/D converter input.

The actual voltage range for a signal going to the A/D converter input is also affected by the 1, 2, 4, 5, 8, 10, 16, and 20 V/V software-programmable gains available on each channel of the ADS7870 A/D converter. Thus, you must scale the analog signal with an attenuator circuit and a software-programmable gain so that the actual input presented to the A/D converter is within the range limits of the ADS7870 A/D converter chip (-2 V to + 2 V or 0 V to + 2 V).

The A/D converter chip can only accept positive voltages. With the R1 resistors connected to ground, your analog circuit is well-suited to perform positive A/D conversions. When the R1 resistors are tied to ground for differential measurements, both differential inputs must be referenced to analog ground, and ***both inputs must be positive with respect to analog ground***.

If a device such as a battery is connected across two channels for a differential measurement, and it is ***not*** referenced to analog ground, then the current from the device will flow through both sets of attenuator resistors without flowing back to analog ground as shown in Figure 11. This will generate a negative voltage at one of the inputs, LN1, which will almost certainly lead to inaccurate A/D conversions. To make such differential measurements, connect the R1 resistors to the A/D converter's internal reference voltage, which is software-configurable for 1.15 V, 2.048 V, or 2.5 V. This internal reference voltage is available on pin 49 of header J3 as VREF, and allows you to convert analog input voltages that are negative with respect to analog ground.

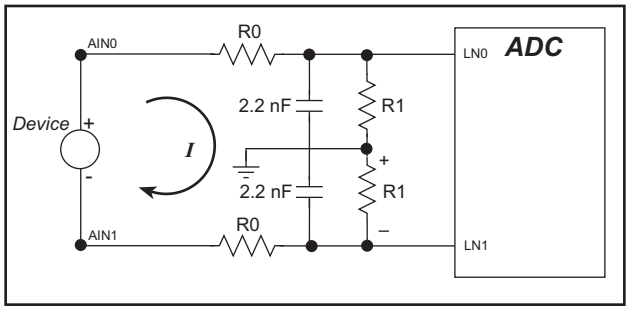


Figure 11. Current Flow from Ungrounded or Floating Source

NOTE: The amplifier inside the A/D converter's internal voltage reference circuit has a very limited output-current capability. The internal buffer can source up to 20 mA and sink only up to 200 µA. Use a separate buffer amplifier if you need to supply any load current.

The A/D converter's CONVERT pin is available on pin 48 of header J3 and can be used as a hardware means of forcing the A/D converter to start a conversion cycle at a specific time. The CONVERT signal is an edge-triggered event and has a hold time of two CCLK periods for debounce.

A conversion is started by an active (rising) edge on the CONVERT pin. The CONVERT pin must stay low for at least two CCLK periods before going high for at least two CCLK periods. Figure 12 shows the timing of a conversion start. The double falling arrow on CCLK indicates the actual start of the conversion cycle.

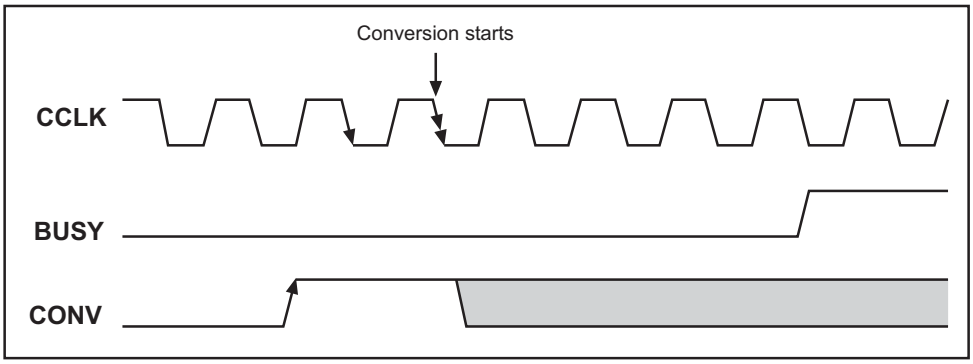


Figure 12. Timing Diagram for Conversion Start Using CONVERT Pin

Appendix B explains the implementation examples of these features on the Prototyping Board.

4.4.1 A/D Converter Power Supply

The analog section is isolated from digital noise generated by other components by way of a low-pass filter composed of C17, L3, and C28 on the RCM4300 as shown in Figure 13. The +V analog power supply powers the A/D converter chip.

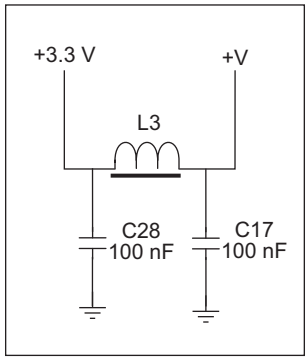


Figure 13. Analog Supply Circuit

4.5 Other Hardware

4.5.1 Clock Doubler

The RCM4300 takes advantage of the Rabbit 4000 microprocessor's internal clock doubler. A built-in clock doubler allows half-frequency crystals to be used to reduce radiated emissions. The 58.98 MHz frequency specified for the RCM4300 model is generated using a 29.49 MHz crystal.

The clock doubler may be disabled if 58.98 MHz clock speeds are not required. Disabling the Rabbit 4000 microprocessor's internal clock doubler will reduce power consumption and further reduce radiated emissions. The clock doubler is disabled with a simple configuration macro as shown below.

1. Select the "Defines" tab from the Dynamic C **Options > Project Options** menu.
2. Add the line **CLOCK_DOUBLED=0** to always disable the clock doubler.

   The clock doubler is enabled by default, and usually no entry is needed. If you need to specify that the clock doubler is always enabled, add the line **CLOCK_DOUBLED=1** to always enable the clock doubler.
3. Click **OK** to save the macro. The clock doubler will now remain off whenever you are in the project file where you defined the macro.

4.5.2 Spectrum Spreader

The Rabbit 4000 features a spectrum spreader, which helps to mitigate EMI problems. The spectrum spreader is on by default, but it may also be turned off or set to a stronger setting. The means for doing so is through a simple configuration macro as shown below.

1. Select the "Defines" tab from the Dynamic C **Options > Project Options** menu.
2. Normal spreading is the default, and usually no entry is needed. If you need to specify normal spreading, add the line

   ```
   ENABLE_SPREADER=1
   ```

   For strong spreading, add the line

   ```
   ENABLE_SPREADER=2
   ```

   To disable the spectrum spreader, add the line

   ```
   ENABLE_SPREADER=0
   ```

   NOTE: The strong spectrum-spreading setting is not recommended since it may limit the maximum clock speed or the maximum baud rate. It is unlikely that the strong setting will be used in a real application.
3. Click **OK** to save the macro. The spectrum spreader will now remain off whenever you are in the project file where you defined the macro.

NOTE: Refer to the ***Rabbit 4000 Microprocessor User's Manual*** for more information on the spectrum-spreading setting and the maximum clock speed.

4.6 Memory

4.6.1 SRAM

All RCM4300 modules have 512K of battery-backed data SRAM installed at U10, and the RCM4300 model has 512K of fast SRAM installed at U12.

4.6.2 Flash Memory

All RCM4300 modules also have up to 2MB of serial flash memory installed at U5.

A "user block" area is defined to store persistent data. The functions **`writeUserBlock`** and **`readUserBlock`** are provided for this. Refer to the ***Rabbit 4000 Microprocessor Designer's Handbook*** for additional information.

4.6.3 VBAT RAM Memory

The tamper detection feature of the Rabbit 4000 microprocessor can be used to detect any attempt to enter the bootstrap mode. When such an attempt is detected, the VBAT RAM memory in the Rabbit 4000 chip is erased. The serial bootloader on RCM4300 RabbitCore modules uses the bootstrap mode to load the SRAM, which erases the VBAT RAM memory on any reset, and so it cannot be used for tamper detection.

4.6.4 *microSD*™ Cards

The RCM4300 supports a removable *microSD*™ *Card* up to a 1 GB to store data and Web pages. The *microSD*™ *Card* is particularly suitable for mass-storage applications, but is generally unsuitable for direct program execution.

Unlike other flash devices, the *microSD*™ *Card* has some intelligence, which facilitates working with it. You do not have to worry about erased pages. All *microSD*™ *Cards* support 512-byte reads and writes, and handle any necessary pre-erasing internally.

Figure 14 shows how to insert or remove the *microSD*™ *Card*. The card is designed to fit easily only one way — do ***not*** bend the card or force it into the slot. While you remove or insert the card, take care to avoid touching the electrical contacts on the bottom of the card to prevent electrostatic discharge damage to the card and to keep any moisture or other contaminants off the contacts. You will sense a soft click once the card is completely inserted. To remove it, gently press the card towards the middle of the RCM4300 — you will sense a soft click and the card will be ready to be removed. Do not attempt to pull the card from the socket before pressing it in — otherwise the ejection mechanism will get damaged. The ejection mechanism is spring-loaded, and will partially eject the card when used correctly.

> **NOTE:** When using the optional Dynamic C FAT file system module, do ***not*** remove or insert the *microSD*™ *Card* while LED DS4 above the *microSD*™ *Card* is on to indicate that the *microSD*™ *Card* is mounted. The LED will go off when the *microSD*™ *Card* is unmounted, indicating that it is safe to remove it.

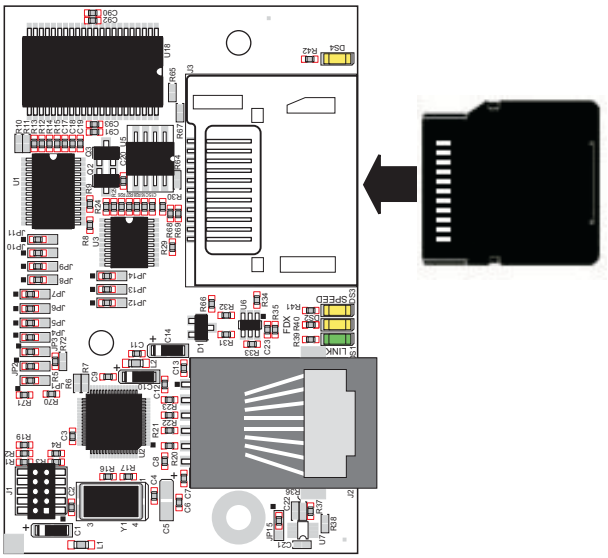


Figure 14. Insertion/Removal of microSD Card

Rabbit Semiconductor recommends that you use the *microSD™ Card* holder at header J3 only for the *microSD™ Card* since other devices are not supported. Be careful to remove and insert the card as shown, and be careful ***not*** to insert any foreign objects, which may short out the contacts and lead to the destruction of your card.

It is possible to hot-swap *microSD™ Card*s without removing power from the RCM4300 modules. The file system must be closed before the cards can be hot-swapped. The chip selects associated with the card must be set to their inactive state, and read/write operations addressed to the *microSD™ Card* port cannot be allowed to occur. These operations can be initiated in software by sensing an external switch actuated by the user, and the card can then be removed and replaced with a different one. Once the application program detects a new card, the file system can be opened. These steps allow the *microSD™ Card* to be installed or removed without affecting either the program, which continues to run on the RCM4300 module, or the data stored on the card. The Dynamic C FAT file system will handle this overhead automatically by unmounting the *microSD™ Card*. LED DS4 above the *microSD™ Card* is used by the FAT file system to show when the media is mounted.

Standard Windows SD Card readers may be used to read the *microSD™ Card* formatted by the Dynamic C FAT file system with the RCM4300 as long as it has not been partitioned. An SD Card adapter is included with the *microSD™ Card* in the in the RCM4300 Development Kit. The SD Card adapter has a sliding switch along the left side that may be moved down to write-protect the *microSD™ Card* while it is being used with an SD Card reader.

Sample programs in the **SAMPLES\RCM4300\SD_Flash** folder illustrate the use of the *microSD™ Card*s. These sample programs are described in Section 3.2.2, “Use of microSD™ Cards,”

5. SOFTWARE REFERENCE

Dynamic C is an integrated development system for writing embedded software. It runs on an IBM-compatible PC and is designed for use with single-board computers and other devices based on the Rabbit microprocessor. Chapter 5 describes the libraries and function calls related to the RCM4300.

5.1 More About Dynamic C

Dynamic C has been in use worldwide since 1989. It is specially designed for programming embedded systems, and features quick compile and interactive debugging. A complete reference guide to Dynamic C is contained in the ***Dynamic C User's Manual***.

Since the RCM4300 has a serial flash memory, all software development must be done in the static SRAM. The flash memory and SRAM options are selected with the **Options > Program Options > Compiler** menu.

> **NOTE:** An application should be compiled directly to the battery-backed data SRAM on the RCM4300 module using the **Code and BIOS in RAM** BIOS memory compiler option while debugging for faster download times, but should be recompiled to run from the fast SRAM after the serial programming cable is disconnected. Your final code must always be stored in flash memory for reliable operation. RCM4300 modules have a fast program execution SRAM that is not battery-backed. Select **Code and BIOS in Flash, Run in RAM** from the Dynamic C **Options > Project Options > Compiler** menu to store the code in flash and copy it to the fast program execution SRAM at runtime to take advantage of the faster clock speed. This option optimizes the performance of RCM4300 modules running at 58.98 MHz.

Developing software with Dynamic C is simple. Users can write, compile, and test C and assembly code without leaving the Dynamic C development environment. Debugging occurs while the application runs on the target. Alternatively, users can compile a program to an image file for later loading. Dynamic C runs on PCs on PCs under Windows NT and later—note that Dynamic C is still being evaluated for compatibility with Windows Vista at the time of writing, and should not be expected to run correctly under Windows Vista at this time. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features.

- Full-feature source and/or assembly-level debugger, no in-circuit emulator required.
- Royalty-free TCP/IP stack with source code and most common protocols.
- Hundreds of functions in source-code libraries and sample programs:
    - Exceptionally fast support for floating-point arithmetic and transcendental functions.
    - RS-232 and RS-485 serial communication.
    - Analog and digital I/O drivers.
    - I^2C, SPI, GPS, file system.
    - LCD display and keypad drivers.
- Powerful language extensions for cooperative or preemptive multitasking
- Loader utility program to load binary images into Rabbit targets in the absence of Dynamic C.
- Provision for customers to create their own source code libraries and augment on-line help by creating "function description" block comments using a special format for library functions.
- Standard debugging features:
    - Breakpoints—Set breakpoints that can disable interrupts.
    - Single-stepping—Step into or over functions at a source or machine code level, µC/OS-II aware.
    - Code disassembly—The disassembly window displays addresses, opcodes, mnemonics, and machine cycle times. Switch between debugging at machine-code level and source-code level by simply opening or closing the disassembly window.
    - Watch expressions—Watch expressions are compiled when defined, so complex expressions including function calls may be placed into watch expressions. Watch expressions can be updated with or without stopping program execution.
    - Register window—All processor registers and flags are displayed. The contents of general registers may be modified in the window by the user.
    - Stack window—shows the contents of the top of the stack.
    - Hex memory dump—displays the contents of memory at any address.
    - **STDIO** window—**`printf`** outputs to this window and keyboard input on the host PC can be detected for debugging purposes. **`printf`** output may also be sent to a serial port or file.

5.2 Dynamic C Function Calls

5.2.1 Digital I/O

The RCM4300 was designed to interface with other systems, and so there are no drivers written specifically for the I/O. The general Dynamic C read and write functions allow you to customize the parallel I/O to meet your specific needs. For example, use

```
WrPortI(PEDDR, &PEDDRShadow, 0x00);
```

to set all the Port E bits as inputs, or use

```
WrPortI(PEDDR, &PEDDRShadow, 0xFF);
```

to set all the Port E bits as outputs.

When using the auxiliary I/O bus on the Rabbit 4000 chip, add the line

```
#define PORTA_AUX_IO    // required to enable auxiliary I/O bus
```

to the beginning of any programs using the auxiliary I/O bus.

The sample programs in the Dynamic C **SAMPLES/RCM4300** folder provide further examples.

5.2.2 Serial Communication Drivers

Library files included with Dynamic C provide a full range of serial communications support. The **RS232.LIB** library provides a set of circular-buffer-based serial functions. The **PACKET.LIB** library provides packet-based serial functions where packets can be delimited by the 9th bit, by transmission gaps, or with user-defined special characters. Both libraries provide blocking functions, which do not return until they are finished transmitting or receiving, and nonblocking functions, which must be called repeatedly until they are finished, allowing other functions to be performed between calls. For more information, see the ***Dynamic C Function Reference Manual*** and Rabbit Semiconductor's Technical Note TN213, ***Rabbit Serial Port Software***.

5.2.3 Serial Flash Memory Use

The RCM4300 module has a serial flash memory that contains the user block and stores the application program. Two function calls are provided to work with the serial boot flash. These function calls are in the Dynamic C **LIB\Rabbit4000\BIOSLIB\BOOTDEV_SFLASH.LIB** library.

sbfRead

```
sbfRead(void *dest, unsigned long offset, unsigned nbytes);
```

DESCRIPTION

Reads up to 64K from anywhere on the serial boot flash. This function call supports both the blocking mode for use with µC/OS-II and a mutex for preemptive multitasking, and the nonblocking mode for cooperative multitasking. See the description for **sbfWriteFlash()** for more information on using a µC/OS-II and a mutex with the serial flash driver.

PARAMETERS

dest	pointer to the 16-bit address of the destination buffer
offset	the physical offset into the serial flash
nbytes	the number of bytes to read

RETURN VALUE

0 if successful.

The return values below apply only if **_SPI_USE_UCOS_MUTEX** is not **#defined**:

positive **N** to indicate that the SPI port is being used by device *n*

if more than **_SPI_MAXTIME** milliseconds elapse while waiting for the SPI port to become available, one of the following two runtime errors will occur: **ERR_SPI_MUTEX_ERROR** (when using µC/OS-II) or **-ETIME** (if not using µC/OS-II).

sbfWriteFlash

```
int sbfWriteFlash(unsigned long flashDst, void* Src,
   unsigned len);
```

DESCRIPTION

Writes **len** bytes (up to 64K) to physical address **flashDst** from **Src**.

Keep calling **sbfWriteFlash()** until it returns zeroa or a negative error code. A positive return value indicates that the serial flash SPI port is being used by another device. If you are using µC/OS-II and **_SPI_USE_UCOS_MUTEX** is **#defined**, you may call **sbfWriteFlash()**just once. If more than **_SPI_MAXTIME** milliseconds elapse while waiting for the SPI port to become available, one of the following two run-time errors will occur: **ERR_SPI_MUTEX_ERROR** (when using µC/OS-II) or **-ETIME** (if not using µC/OS-II).

NOTE: This function call is ***not*** power-fail safe. The **writeUserBlock()** function call provides a safer way to store critical data using redundant copies.

PARAMETERS

flashDst	the physical address of the flash destination
Src	near pointer to the source dat5a
len	the number of bytes to write

RETURN VALUE

0 if successful.

-1 if an attempt was made to write to the user/ID block or program area.

The return values below apply only if **_SPI_USE_UCOS_MUTEX** is not **#defined**:

-EBUSY to indicate a busy writing to the serial flash

positive **N** to indicate that the SPI port is being used by device *n*

if more than **_SPI_MAXTIME** milliseconds elapse while waiting for the SPI port to become available, one of the following two runtime errors will occur: **ERR_SPI_MUTEX_ERROR** (when using µC/OS-II) or **-ETIME** (if not using µC/OS-II).

5.2.4 User Block

Certain function calls involve reading and storing calibration constants from/to the simulated EEPROM in flash memory located at the top 2K of the reserved user block memory area (3800–39FF). This leaves the address range 0–37FF in the user block available for your application.

These address ranges may change in the future in response to the volatility in the flash memory market, in particular sector size. The sample program **USERBLOCK_INFO.C** in the Dynamic C **SAMPLES\USERBLOCK** folder can be used to determine the version of the ID block, the size of the ID and user blocks, whether or not the ID/user blocks are mirrored, the total amount of flash memory used by the ID and user blocks, and the area of the user block available for your application.

The **USERBLOCK_CLEAR.C** sample program shows you how to clear and write the contents of the user block that you are using in your application (the calibration constants in the reserved area and the ID block are protected).

> **NOTE:** Since RCM4300 RabbitCore modules have a serial boot flash that shares the serial flash SPI lines with other devices, exercise care when accessing the user block. Pay attention to the instructions associated with the user block function calls in the Dynamic C **LIB\Rabbit4000\BIOSLIB\IDBLOCK.LIB** library.

5.2.5 SRAM Use

The RCM4300 module has a battery-backed data SRAM and a program-execution SRAM. Dynamic C provides the **protected** keyword to identify variables that are to be placed into the battery-backed SRAM. The compiler generates code that maintains two copies of each protected variable in the battery-backed SRAM. The compiler also generates a flag to indicate which copy of the protected variable is valid at the current time. This flag is also stored in the battery-backed SRAM. When a protected variable is updated, the "inactive" copy is modified, and is made "active" only when the update is 100% complete. This assures the integrity of the data in case a reset or a power failure occurs during the update process. At power-on the application program uses the active copy of the variable pointed to by its associated flag.

The sample code below shows how a protected variable is defined and how its value can be restored.

```
main() {
   protected int state1, state2, state3;
   ...
   _sysIsSoftReset();    // restore any protected variables
```

The **bbram** keyword may also be used instead if there is a need to store a variable in battery-backed SRAM without affecting the performance of the application program. Data integrity is ***not*** assured when a reset or power failure occurs during the update process.

Additional information on **bbram** and **protected** variables is available in the ***Dynamic C User's Manual***.

5.2.6 RCM4300 Cloning

The RCM4300 does not have a pull-up resistor on the PB1 (CLKA) line of the programming port. Because of this, the procedure to generate clones from the RCM4300 differs from that used for other RabbitCore modules and single-boards computers. You must set the **CL_FORCE_MASTER_MODE** macro to 1 in the Dynamic C **LIB\Rabbit4000\BIOSLIB\CLONECONFIG.LIB** library to use the RCM4300 as a master for cloning. An RCM4300 master will not run the application, and further debugging is not possible as long as the **CL_FORCE_MASTER_MODE** macro is set to 1. Any cloned RCM4300 modules will be "sterile," meaning that they cannot be used as a master for cloning. To develop and debug an application on an RCM4300, comment out the **CL_FORCE_MASTER_MODE** macro or set it to 0.

> **NOTE:** Instead of defining this macro is your application, you may simply add the line **CL_FORCE_MASTER_MODE=1** under the Dynamic C **Options > Project Options** "Defines" tab, then click **OK**. When you recompile your program, this will have the same effect as setting the macro to 1 within the **CLONECONFIG.LIB** library.

See Rabbit Semiconductor's Technical Note TN207, ***Rabbit Cloning Board***, for additional information on Rabbit Semiconductor's cloning board and how cloning is done.

5.2.7 *microSD*™ Card Drivers

The Dynamic C **LIB\Rabbit4000\SDflash\SDFLASH.LIB** library is used to interface to *microSD*™ *Card* memory devices on an SPI bus. More information on these function calls is available in the ***Dynamic C Function Reference Manual***.

The *microSD*™ *Card* is ideally suited to store files with a directory structure. The Dynamic C FAT file system module provides support for a file system and for formatting the *microSD*™ *Card* for use in a Rabbit-based system. This allows files to be read and written in a PC-compatible manner. Visit our Web site at www.rabbit.com or contact your Rabbit Semiconductor sales representative or authorized distributor for further information on the Dynamic C FAT File System and other Dynamic C modules. The supporting documentation for the Dynamic C FAT File System and the sample programs in the **SAMPLES\FileSystem\FAT** folder illustrate the use of the Dynamic C FAT file system.

5.2.8 Prototyping Board Function Calls

The functions described in this section are for use with the Prototyping Board features. The source code is in the Dynamic C **LIB\Rabbit4000\RCM4xxx\RCM43xx.LIB** library if you need to modify it for your own board design.

There are several internal function calls (denoted by an underscore at the start of the function name) that should not be modified. These internal function calls control the SPI port sharing.

> **NOTE:** The analog input function calls are supported only by the RCM4300 model since the RCM4310 does not have an A/D converter.

The sample programs in the Dynamic C **SAMPLES\RCM4300** folder illustrate the use of the function calls.

Other generic functions applicable to all devices based on Rabbit microprocessors are described in the ***Dynamic C Function Reference Manual***.

5.2.8.1 Board Initialization

---

brdInit

```
void brdInit(void);
```

DESCRIPTION

Call this function at the beginning of your program. This function initializes Parallel Ports A through E for use with the Prototyping Board, and on the RCM4300 model loads the stored calibration constants for the A/D converter. This function call is intended for demonstration purposes only, and can be modified for your applications.

Summary of Initialization

1. I/O port pins are configured for Prototyping Board operation.
2. Unused configurable I/O are set as tied outputs.
3. RS-232 is not enabled.
4. LEDs are off.
5. The slave port is disabled.

RETURN VALUE

None.

5.2.8.2 Alerts

These function calls can be found in the Dynamic C **LIB\Rabbit4000\RCM4xxx\RCM4xxx.LIB** library.

timedAlert

```
void timedAlert(unsigned long timeout);
```

DESCRIPTION

Polls the real-time clock until a timeout occurs. The RCM4400W will be in a low-power mode during this time. Once the timeout occurs, this function call will enable the normal power source.

PARAMETER

timeout	the duration of the timeout in seconds

RETURN VALUE

None.

digInAlert

```
void digInAlert(int dataport, int portbit, int value,
   unsigned long timeout);
```

DESCRIPTION

Polls a digital input for a set value or until a timeout occurs. The RCM4400W will be in a low-power mode during this time. Once a timeout occurs or the correct byte is received, this function call will enable the normal power source and exit.

PARAMETERS

dataport	the input port data register to poll (e.g., PADR)
portbit	the input port bit (0–7) to poll
value	the value of 0 or 1 to receive
timeout	the duration of the timeout in seconds (enter 0 for no timeout)

RETURN VALUE

None.

5.2.9 Analog Inputs (RCM4300 only)

The function calls used with the Prototyping Board features and the A/D converter on the RCM4300 model are in the Dynamic C **LIB\Rabbit4000\RCM4xxx\ADC_ADS7870.LIB** library. Dynamic C v. 10.07 or later is required to use the A/D converter function calls.

anaInConfig

```
unsigned int anaInConfig(unsigned int instructionbyte,
   unsigned int cmd, long brate);
```

DESCRIPTION

Use this function to configure the A/D converter. This function will address the A/D converter chip in Register Mode only, and will return an error if you try the Direct Mode. Appendix B.4.3 provides additional addressing and command information.

ADS7870 Signal	ADS7870 State	RCM4300 Function/State
LN0	Input	AIN0
LN1	Input	AIN1
LN2	Input	AIN2
LN3	Input	AIN3
LN4	Input	AIN4
LN5	Input	AIN5
LN6	Input	AIN6
LN7	Input	AIN7
/RESET	Input	Board reset device
RISE/FALL	Input	Pulled up for SCLK active on rising edge
I/O0	Input	Pulled down
I/O1	Input	Pulled down
I/O2	Input	Pulled down
I/O3	Input	Pulled down
CONVERT	Input	Pulled down when not driven
BUSY	Output	PE0 pulled down; logic high state converter is busy
CCLKCNTRL	Input	Pulled down; 0 state sets CCLK as input
CCLK	Input	Pulled down; external conversion clock
SCLK	Input	PB0; serial data transfer clock
SDI	Input	PC4; 3-wire mode for serial data input
SDO	Output	PC5; serial data output /CS driven
/CS	Input	BUFEN pulled up; active-low enables serial interface
BUFIN	Input	Driven by VREF
VREF	Output	Connected to BUFIN and BUFOUT
BUFOUT	Output	Driven by VREF

PARAMETERS

instructionbyte the instruction byte that will initiate a read or write operation at 8 or 16 bits on the designated register address. For example,

```
checkid = anaInConfig(0x5F, 0, 9600);
   // read ID and set baud rate
```

cmd the command data that configure the registers addressed by the instruction byte. Enter 0 if you are performing a read operation. For example,

```
i = anaInConfig(0x07, 0x3b, 0);
   // write ref/osc reg and enable
```

brate the serial clock transfer rate of 9600 to 115,200 bytes per second. **brate** must be set the first time this function is called. Enter 0 for this parameter thereafter, for example,

```
anaInConfig(0x00, 0x00, 9600);
   // resets device and sets byte rate
```

RETURN VALUE

0 on write operations.

data value on read operations.

ADSPIBUSY (-4094) if the SPI port is already in use (if more than **_SPI_MAXTIME** milliseconds elapse since the last attempt to grab the port semaphore by the A/D converter, a fatal runtime error, **-ETIME**, results).

SEE ALSO

anaInDriver, **anaIn**, **brdInit**

anaInDriver

```
int anaInDriver(unsigned int cmd);
```

DESCRIPTION

Reads the voltage of an analog input channel by serial-clocking an 8-bit command to the A/D converter by its Direct Mode method. This function assumes that Mode1 (most significant byte first) and the A/D converter oscillator have been enabled. See **anaInConfig()** for the setup.

The conversion begins immediately after the last data bit has been transferred. An exception error will occur if Direct Mode bit D7 is not set.

PARAMETERS

cmd contains a gain code and a channel code as follows.

D7—1; D6–D4—Gain Code; D3–D0—Channel Code

Use the following calculation and the tables below to determine **cmd**:

```
cmd = 0x80 | (gain_code*16) + channel_code
```

Gain Code	Actual Gain
0	1
1	1.8
2	3.6
3	4.5
4	7.2
5	9.0
6	14.4
7	18

Channel Code	Differential Input Lines	Channel Code	Single-Ended Input Lines*	4–20 mA Lines
0	+AIN0 -AIN1	8	AIN0	AIN0*
1	+AIN2 -AIN3	9	AIN1	AIN1*
2	+AIN4 -AIN5	10	AIN2	AIN2*
3†	+AIN6 -AIN7	11	AIN3	AIN3
4	-AIN0 +AIN1	12	AIN4	AIN4
5	-AIN2 +AIN3	13	AIN5	AIN5
6	-AIN4 +AIN5	14	AIN6	AIN6
7‡	-AIN6 +AIN7	15	AIN7	AIN7*

* Negative input is ground.
† Not accessible on Prototyping Board
‡ Not accessible on Prototyping Board

RETURN VALUE

A value corresponding to the voltage on the analog input channel:

0–2047 for 11-bit conversions.

-2048–2047 for 12-bit conversions.

ADSPIBUSY (-4094) if the A/D converter is locked out or if the SPI port is in use (if more than **_SPI_MAXTIME** milliseconds elapse since the last attempt to grab the port semaphore by the A/D converter, a fatal runtime error, **-ETIME**, results).

ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout.

ADOVERFLOW (-4096) for overflow or out of range.

SEE ALSO

anaInConfig, **anaIn**, **brdInit**

anaIn

```
int anaIn(unsigned int channel, int opmode, int gaincode);
```

DESCRIPTION

Reads the value of an analog input channel using the Direct Mode method of addressing the A/D converter. Note that it takes about 1 second to ensure an internal capacitor on the A/D converter is charged when the function is called the first time.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

opmode the mode of operation:

SINGLE—single-ended input
DIFF—differential input
mAMP—4–20 mA input

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*

* Not accessible on Prototyping Board.

gaincode the gain code of 0 to 7 (applies only to Prototyping Board):

Gain Code	Actual Gain	Voltage Range (V)
0	1	0–22.5
1	1.8	0–11.25
2	3.6	0–5.6
3	4.5	0–4.5
4	7.2	0–2.8
5	9.0	0–2.25
6	14.4	0–1.41
7	18	0–1.126

RETURN VALUE

A value corresponding to the voltage on the analog input channel:

0–2047 for single-ended conversions.

-2048–2047 for differential conversions.

ADSPIBUSY (-4094) if the A/D converter is locked out or if the SPI port is in use (if more than **_SPI_MAXTIME** milliseconds elapse since the last attempt to grab the port semaphore by the A/D converter, a fatal runtime error, **-ETIME**, results).

ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout.

ADOVERFLOW (-4096) for overflow or out of range.

SEE ALSO

anaIn, **anaInConfig**, **anaInDriver**

anaInCalib

```
int anaInCalib(int channel, int opmode, int gaincode,
   int value1, float volts1, int value2, float volts2);
```

DESCRIPTION

Calibrates the response of the desired A/D converter channel as a linear function using the two conversion points provided. Four values are calculated and placed into global tables **_adcCalibS**, **_adcCalibD**, and **adcCalibM** to be later stored into simulated EEPROM using the function **anaInEEWr()**. Each channel will have a linear constant and a voltage offset.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

opmode the mode of operation:

SINGLE—single-ended input
DIFF—differential input
mAMP—4–20 mA input

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*

* Not accessible on Prototyping Board.

gaincode the gain code of 0 to 7 (applies only to Prototyping Board):

Gain Code	Actual Gain	Voltage Range (V)
0	1	0–22.5
1	1.8	0–11.25
2	3.6	0–5.6
3	4.5	0–4.5
4	7.2	0–2.8
5	9.0	0–2.25
6	14.4	0–1.41
7	18	0–1.126

value1 the first A/D converter channel raw count value (0–2047)

volts1 the voltage or current corresponding to the first A/D converter channel value (0 to +20 V or 4 to 20 mA)

value2 the second A/D converter channel raw count value (0–2047)

volts2 the voltage or current corresponding to the first A/D converter channel value (0 to +20 V or 4 to 20 mA)

RETURN VALUE

0 if successful.

-1 if not able to make calibration constants.

SEE ALSO

anaIn, **anaInVolts**, **anaInmAmps**, **anaInDiff**, **anaInCalib**, **brdInit**

anaInVolts

```
float anaInVolts(unsigned int channel, unsigned int gaincode);
```

DESCRIPTION

Reads the state of a single-ended analog input channel and uses the previously set calibration constants to convert it to volts.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

Channel Code	Single-Ended Input Lines*	Voltage Range† (V)
0	+AIN0	0–22.5
1	+AIN1	0–22.5
2	+AIN2	0–22.5
3	+AIN3	0–22.5
4	+AIN4	0–22.5
5	+AIN5	0–22.5
6	+AIN6	0–22.5
7	+AIN7	0–2‡

* Negative input is ground.
† Applies to Prototyping Board.
‡ Used for thermistor in sample program.

gaincode the gain code of 0 to 7 (applies only to Prototyping Board):

Gain Code	Actual Gain	Voltage Range (V)
0	1	0–22.5
1	1.8	0–11.25
2	3.6	0–5.6
3	4.5	0–4.5
4	7.2	0–2.8
5	9.0	0–2.25
6	14.4	0–1.41
7	18	0–1.126

RETURN VALUE

A voltage value corresponding to the voltage on the analog input channel.

ADSPIBUSY (-4094) if the A/D converter is locked out or if the SPI port is in use (if more than **_SPI_MAXTIME** milliseconds elapse since the last attempt to grab the port semaphore by the A/D converter, a fatal runtime error, **-ETIME**, results).

ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout.

ADOVERFLOW (-4096) for overflow or out of range.

SEE ALSO

anaInCalib, **anaIn**, **anaInmAmps**, **brdInit**

anaInDiff

```
float anaInDiff(unsigned int channel, unsigned int gaincode);
```

DESCRIPTION

Reads the state of differential analog input channels and uses the previously set calibration constants to convert it to volts.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

channel	DIFF	Voltage Range (V)
0	+AIN0 -AIN1	-22.5 to +22.5*
1	+AIN1 -AIN1	—
2	+AIN2 -AIN3	-22.5 to +22.5*
3	+AIN3 -AIN3	—
4	+AIN4 -AIN5	-22.5 to +22.5*
5	+AIN5 -AIN5	—
6	+AIN6 -AIN7	—
7	+AIN7 -AIN7	—

* Accessible on Prototyping Board.

gaincode the gain code of 0 to 7 (applies only to Prototyping Board):

Gain Code	Actual Gain	Voltage Range (V)
0	1	-22.5 – +22.5
1	1.8	-11.25 – +11.25
2	3.6	-5.6 – +5.6
3	4.5	-4.5 – +4.5
4	7.2	-2.8 – +2.8
5	9.0	-2.25 – +2.25
6	14.4	-1.41 – +1.41
7	18	-1.126 – +1.126

RETURN VALUE

A voltage value corresponding to the voltage differential on the analog input channel.

ADSPIBUSY (-4094) if the A/D converter is locked out or if the SPI port is in use (if more than **_SPI_MAXTIME** milliseconds elapse since the last attempt to grab the port semaphore by the A/D converter, a fatal runtime error, **-ETIME**, results).

ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout.

ADOVERFLOW (-4096) for overflow or out of range.

SEE ALSO

anaInCalib, **anaIn**, **anaInmAmps**, **brdInit**

anaInmAmps

```
float anaInmAmps(unsigned int channel);
```

DESCRIPTION

Reads the state of an analog input channel and uses the previously set calibration constants to convert it to current.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

Channel Code	4–20 mA Input Lines*
0	+AIN0
1	+AIN1
2	+AIN2
3	+AIN3†
4	+AIN4*
5	+AIN5*
6	+AIN6*
7	+AIN7

* Negative input is ground.

† Applies to Prototyping Board.

RETURN VALUE

A current value between 4.00 and 20.00 mA corresponding to the current on the analog input channel.

ADSPIBUSY (-4094) if the A/D converter is locked out or if the SPI port is in use (if more than **_SPI_MAXTIME** milliseconds elapse since the last attempt to grab the port semaphore by the A/D converter, a fatal runtime error, **-ETIME**, results).

ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout.

ADOVERFLOW (-4096) for overflow or out of range.

SEE ALSO

anaInCalib, **anaIn**, **anaInVolts**

anaInEERd

```
root int anaInEERd(unsigned int channel, unsigned int opmode,
   unsigned int gaincode);
```

DESCRIPTION

Reads the calibration constants, gain, and offset for an input based on their designated position in the flash memory, and places them into global tables **_adcCalibS**, **_adcCalibD**, and **_adcCalibM** for analog inputs. Depending on the flash size, the following macros can be used to identify the starting address for these locations.

ADC_CALIB_ADDRS, address start of single-ended analog input channels

ADC_CALIB_ADDRD, address start of differential analog input channels

ADC_CALIB_ADDRM, address start of milliamp analog input channels

NOTE: This function cannot be run in RAM.

PARAMETER

channel the channel number (0 to 7) corresponding to LN0 to LN7.

opmode the mode of operation:

SINGLE—single-ended input
DIFF—differential input
mAMP—4–20 mA input

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*
ALLCHAN	read all channels for selected **opmode**		

* Not accessible on Prototyping Board.

gaincode the gain code of 0 to 7. The **gaincode** parameter is ignored when **channel** is **ALLCHAN**.

Gain Code	Actual Gain	Voltage Range* (V)
0	1	0–22.5
1	1.8	0–11.25
2	3.6	0–5.6
3	4.5	0–4.5
4	7.2	0–2.8
5	9.0	0–2.25
6	14.4	0–1.41
7	18	0–1.126

* Applies to Prototyping Board.

RETURN VALUE

0 if successful.
-1 if address is invalid or out of range.

SEE ALSO

anaInEEWr, **anaInCalib**

anaInEEWr

```
int anaInEEWr(unsigned int channel, int opmode,
   unsigned int gaincode);
```

DESCRIPTION

Writes the calibration constants, gain, and offset for an input based from global tables **_adcCalibS**, **_adcCalibD**, and **_adcCalibM** to designated positions in the flash memory. Depending on the flash size, the following macros can be used to identify the starting address for these locations.

ADC_CALIB_ADDRS, address start of single-ended analog input channels

ADC_CALIB_ADDRD, address start of differential analog input channels

ADC_CALIB_ADDRM, address start of milliamp analog input channels

NOTE: This function cannot be run in RAM.

PARAMETER

channel the channel number (0 to 7) corresponding to LN0 to LN7.

opmode the mode of operation:

SINGLE—single-ended input
DIFF—differential input
mAMP—4–20 mA input

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*
ALLCHAN	read all channels for selected **opmode**		

* Not accessible on Prototyping Board.

gaincode the gain code of 0 to 7. The **gaincode** parameter is ignored when **channel** is **ALLCHAN**.

Gain Code	Actual Gain	Voltage Range* (V)
0	1	0–22.5
1	1.8	0–11.25
2	3.6	0–5.6
3	4.5	0–4.5
4	7.2	0–2.8
5	9.0	0–2.25
6	14.4	0–1.41
7	18	0–1.126

* Applies to Prototyping Board.

RETURN VALUE

0 if successful
-1 if address is invalid or out of range.

SEE ALSO

anaInEEWr, **anaInCalib**

5.3 Upgrading Dynamic C

Dynamic C patches that focus on bug fixes are available from time to time. Check the Web site www.rabbit.com/support/ for the latest patches, workarounds, and bug fixes.

5.3.1 Add-On Modules

Dynamic C installations are designed for use with the board they are included with, and are included at no charge as part of our low-cost kits. Rabbit Semiconductor offers for purchase add-on Dynamic C modules including the popular μC/OS-II real-time operating system, as well as PPP, Advanced Encryption Standard (AES), FAT file system, Rabbit-Web, and other select libraries.

> **NOTE:** Version 2.13 or later of the Dynamic C FAT file system module is required for the RCM4300 modules.

Each Dynamic C add-on module has complete documentation and sample programs to illustrate the functionality of the software calls in the module. Visit our Web site at www.rabbit.com for further information and complete documentation for each module.

In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support module is also available for purchase.

6. USING THE TCP/IP FEATURES

6.1 TCP/IP Connections

Programming and development can be done with the RCM4300 without connecting the Ethernet port to a network. However, if you will be running the sample programs that use the Ethernet capability or will be doing Ethernet-enabled development, you should connect the RCM4300 module's Ethernet port at this time.

Before proceeding you will need to have the following items.

- If you don't have Ethernet access, you will need at least a 10Base-T Ethernet card (available from your favorite computer supplier) installed in a PC.
- Two RJ-45 straight-through Ethernet cables and a hub, or an RJ-45 crossover Ethernet cable.

Figure 15 shows how to identify the two Ethernet cables based on the wires in the transparent RJ-45 connectors.

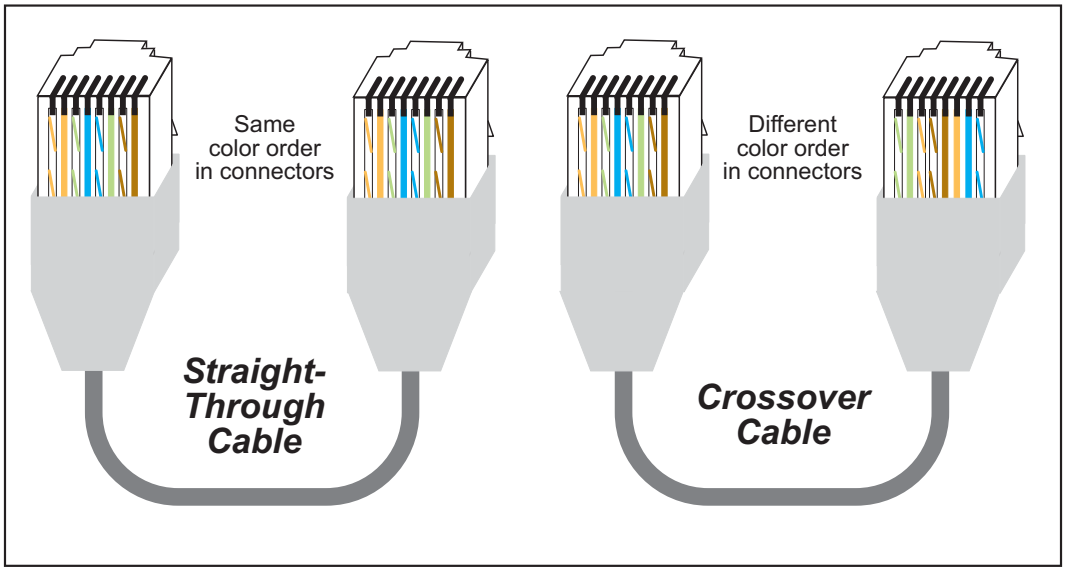


Figure 15. How to Identify Straight-Through and Crossover Ethernet Cables

Ethernet cables and a 10Base-T Ethernet hub are available from Rabbit Semiconductor in a TCP/IP tool kit. More information is available at www.rabbit.com.

Now you should be able to make your connections.

1. Connect the AC adapter and the serial programming cable as shown in Chapter 2, "Getting Started."
2. Ethernet Connections

   There are four options for connecting the RCM4300 module to a network for development and runtime purposes. The first two options permit total freedom of action in selecting network addresses and use of the "network," as no action can interfere with other users. We recommend one of these options for initial development.

- **No LAN —** The simplest alternative for desktop development. Connect the RCM4300 module's Ethernet port directly to the PC's network interface card using an RJ-45 *crossover cable*. A crossover cable is a special cable that flips some connections between the two connectors and permits direct connection of two client systems. A standard RJ-45 network cable will not work for this purpose.
- **Micro-LAN —** Another simple alternative for desktop development. Use a small Ethernet 10Base-T hub and connect both the PC's network interface card and the RCM4300 module's Ethernet port to it using standard network cables.

The following options require more care in address selection and testing actions, as conflicts with other users, servers and systems can occur:

- **LAN —** Connect the RCM4300 module's Ethernet port to an existing LAN, preferably one to which the development PC is already connected. You will need to obtain IP addressing information from your network administrator.
- **WAN —** The RCM4300 is capable of direct connection to the Internet and other Wide Area Networks, but exceptional care should be used with IP address settings and all network-related programming and development. We recommend that development and debugging be done on a local network before connecting a RabbitCore system to the Internet.

  TIP: Checking and debugging the initial setup on a micro-LAN is recommended before connecting the system to a LAN or WAN.

The PC running Dynamic C does not need to be the PC with the Ethernet card.

3. Apply Power

   Plug in the AC adapter. The RCM4300 module and Prototyping Board are now ready to be used.

6.2 TCP/IP Primer on IP Addresses

Obtaining IP addresses to interact over an existing, operating, network can involve a number of complications, and must usually be done with cooperation from your ISP and/or network systems administrator. For this reason, it is suggested that the user begin instead by using a direct connection between a PC and the RCM4300 using an Ethernet crossover cable or a simple arrangement with a hub. (A crossover cable should not be confused with regular straight through cables.)

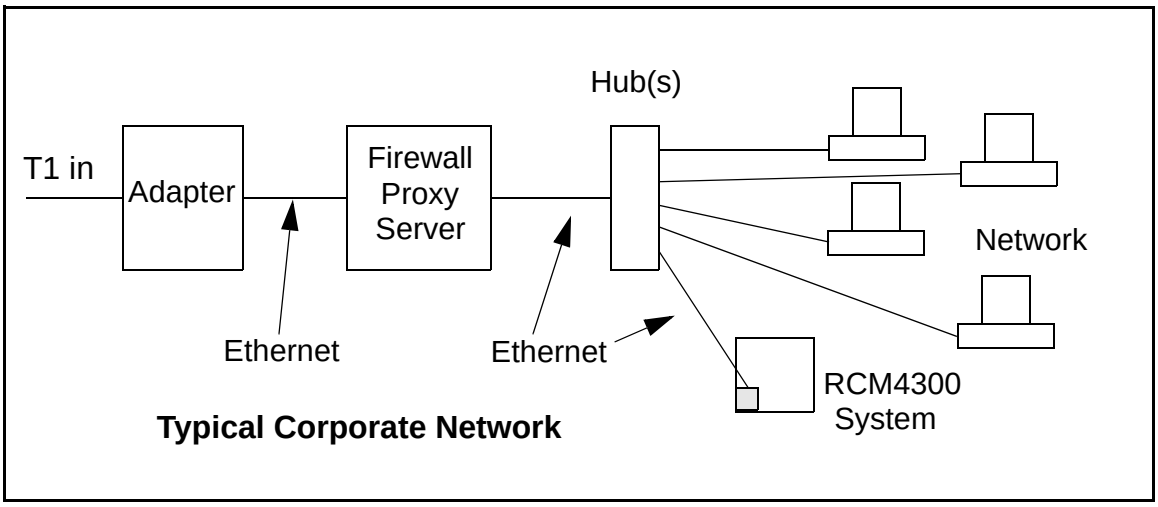
In order to set up this direct connection, the user will have to use a PC without networking, or disconnect a PC from the corporate network, or install a second Ethernet adapter and set up a separate private network attached to the second Ethernet adapter. Disconnecting your PC from the corporate network may be easy or nearly impossible, depending on how it is set up. If your PC boots from the network or is dependent on the network for some or all of its disks, then it probably should not be disconnected. If a second Ethernet adapter is used, be aware that Windows TCP/IP will send messages to one adapter or the other, depending on the IP address and the binding order in Microsoft products. Thus you should have different ranges of IP addresses on your private network from those used on the corporate network. If both networks service the same IP address, then Windows may send a packet intended for your private network to the corporate network. A similar situation will take place if you use a dial-up line to send a packet to the Internet. Windows may try to send it via the local Ethernet network if it is also valid for that network.

The following IP addresses are set aside for local networks and are not allowed on the Internet: 10.0.0.0 to 10.255.255.255, 172.16.0.0 to 172.31.255.255, and 192.168.0.0 to 192.168.255.255.

The RCM4300 uses a 10Base-T type of Ethernet connection, which is the most common scheme. The RJ-45 connectors are similar to U.S. style telephone connectors, except they are larger and have 8 contacts.

An alternative to the direct connection using a crossover cable is a direct connection using a hub. The hub relays packets received on any port to all of the ports on the hub. Hubs are low in cost and are readily available. The RCM4300 uses 10 Mbps Ethernet, so the hub or Ethernet adapter can be a 10 Mbps unit or a 10/100 Mbps unit.

In a corporate setting where the Internet is brought in via a high-speed line, there are typically machines between the outside Internet and the internal network. These machines include a combination of proxy servers and firewalls that filter and multiplex Internet traffic. In the configuration below, the RCM4300 could be given a fixed address so any of the computers on the local network would be able to contact it. It may be possible to configure the firewall or proxy server to allow hosts on the Internet to directly contact the controller, but it would probably be easier to place the controller directly on the external network outside of the firewall. This avoids some of the configuration complications by sacrificing some security.



Typical Corporate Network

If your system administrator can give you an Ethernet cable along with its IP address, the netmask and the gateway address, then you may be able to run the sample programs without having to setup a direct connection between your computer and the RCM4300. You will also need the IP address of the nameserver, the name or IP address of your mail server, and your domain name for some of the sample programs.

6.2.1 IP Addresses Explained

IP (Internet Protocol) addresses are expressed as 4 decimal numbers separated by periods, for example:

216.103.126.155

10.1.1.6

Each decimal number must be between 0 and 255. The total IP address is a 32-bit number consisting of the 4 bytes expressed as shown above. A local network uses a group of adjacent IP addresses. There are always 2^N IP addresses in a local network. The netmask (also called subnet mask) determines how many IP addresses belong to the local network. The netmask is also a 32-bit address expressed in the same form as the IP address. An example netmask is:

255.255.255.0

This netmask has 8 zero bits in the least significant portion, and this means that 2^8 addresses are a part of the local network. Applied to the IP address above (216.103.126.155), this netmask would indicate that the following IP addresses belong to the local network:

216.103.126.0

216.103.126.1

216.103.126.2

etc.

216.103.126.254

216.103.126.255

The lowest and highest address are reserved for special purposes. The lowest address (216.102.126.0) is used to identify the local network. The highest address (216.102.126.255) is used as a broadcast address. Usually one other address is used for the address of the gateway out of the network. This leaves 256 - 3 = 253 available IP addresses for the example given.

6.2.2 How IP Addresses are Used

The actual hardware connection via an Ethernet uses Ethernet adapter addresses (also called MAC addresses). These are 48-bit addresses and are unique for every Ethernet adapter manufactured. In order to send a packet to another computer, given the IP address of the other computer, it is first determined if the packet needs to be sent directly to the other computer or to the gateway. In either case, there is an Ethernet address on the local network to which the packet must be sent. A table is maintained to allow the protocol driver to determine the MAC address corresponding to a particular IP address. If the table is empty, the MAC address is determined by sending an Ethernet broadcast packet to all devices on the local network asking the device with the desired IP address to answer with its MAC address. In this way, the table entry can be filled in. If no device answers, then the device is nonexistent or inoperative, and the packet cannot be sent.

Some IP address ranges are reserved for use on internal networks, and can be allocated freely as long as no two internal hosts have the same IP address. These internal IP addresses are not routed to the Internet, and any internal hosts using one of these reserved IP addresses cannot communicate on the external Internet without being connected to a host that has a valid Internet IP address. The host would either translate the data, or it would act as a proxy.

Each RCM4300 RabbitCore module has its own unique MAC address, which consists of the prefix 0090C2 followed by a code that is unique to each RCM4300 module. For example, a MAC address might be 0090C2C002C0.

> **TIP:** You can always obtain the MAC address on your module by running the sample program **DISPLAY_MAC.C** from the **SAMPLES\TCPIP** folder.

6.2.3 Dynamically Assigned Internet Addresses

In many instances, devices on a network do not have fixed IP addresses. This is the case when, for example, you are assigned an IP address dynamically by your dial-up Internet service provider (ISP) or when you have a device that provides your IP addresses using the Dynamic Host Configuration Protocol (DHCP). The RCM4300 modules can use such IP addresses to send and receive packets on the Internet, but you must take into account that this IP address may only be valid for the duration of the call or for a period of time, and could be a private IP address that is not directly accessible to others on the Internet. These addresses can be used to perform some Internet tasks such as sending e-mail or browsing the Web, but it is more difficult to participate in conversations that originate elsewhere on the Internet. If you want to find out this dynamically assigned IP address, under Windows NT or later you can run the **ipconfig** command (**Start > Run >cmd**) while you are connected and look at the interface used to connect to the Internet.

Many networks use IP addresses that are assigned using DHCP. When your computer comes up, and periodically after that, it requests its networking information from a DHCP server. The DHCP server may try to give you the same address each time, but a fixed IP address is usually not guaranteed.

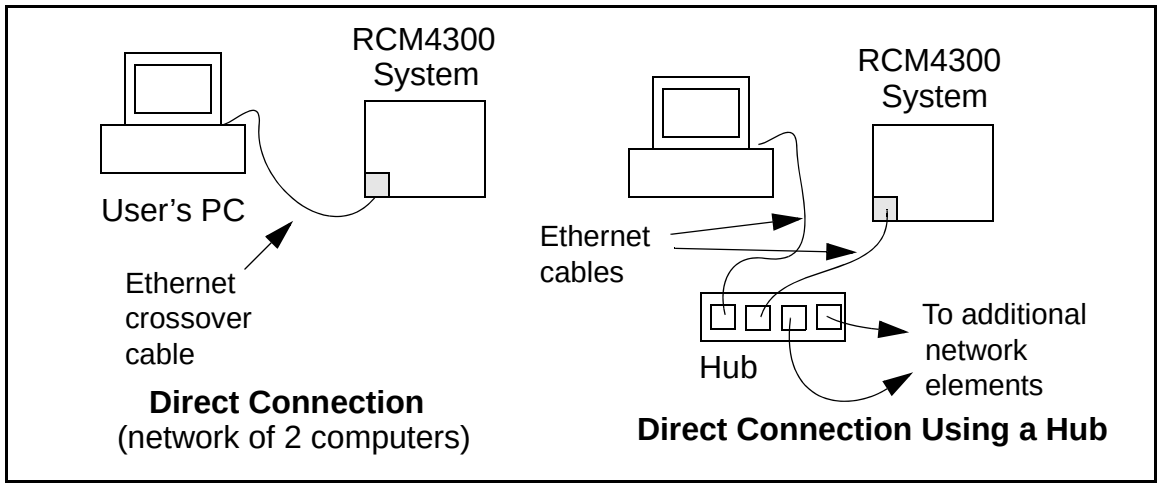
If you are not concerned about accessing the RCM4300 from the Internet, you can place the RCM4300 on the internal network using an IP address assigned either statically or through DHCP.

6.3 Placing Your Device on the Network

In many corporate settings, users are isolated from the Internet by a firewall and/or a proxy server. These devices attempt to secure the company from unauthorized network traffic, and usually work by disallowing traffic that did not originate from inside the network. If you want users on the Internet to communicate with your RCM4300, you have several options. You can either place the RCM4300 directly on the Internet with a real Internet address or place it behind the firewall. If you place the RCM4300 behind the firewall, you need to configure the firewall to translate and forward packets from the Internet to the RCM4300.

6.4 Running TCP/IP Sample Programs

We have provided a number of sample programs demonstrating various uses of TCP/IP for networking embedded systems. These programs require you to connect your PC and the RCM4300 module together on the same network. This network can be a local private network (preferred for initial experimentation and debugging), or a connection via the Internet.



6.4.1 How to Set IP Addresses in the Sample Programs

With the introduction of Dynamic C 7.30 we have taken steps to make it easier to run many of our sample programs. You will see a **TCPCONFIG** macro. This macro tells Dynamic C to select your configuration from a list of default configurations. You will have three choices when you encounter a sample program with the **TCPCONFIG** macro.

1. You can replace the **TCPCONFIG** macro with individual **MY_IP_ADDRESS**, **MY_NETMASK**, **MY_GATEWAY**, and **MY_NAMESERVER** macros in each program.
2. You can leave **TCPCONFIG** at the usual default of 1, which will set the IP configurations to **10.10.6.100**, the netmask to **255.255.255.0**, and the nameserver and gateway to **10.10.6.1**. If you would like to change the default values, for example, to use an IP address of **10.1.1.2** for the RCM4300 module, and **10.1.1.1** for your PC, you can edit the values in the section that directly follows the "General Configuration" comment in the **TCP_CONFIG.LIB** library. You will find this library in the **LIB\Rabbit4000\TCPIP** directory.
3. You can create a **CUSTOM_CONFIG.LIB** library and use a **TCPCONFIG** value greater than 100. Instructions for doing this are at the beginning of the **TCP_CONFIG.LIB** library in the **LIB\Rabbit4000\TCPIP** directory.

There are some other "standard" configurations for **TCPCONFIG** that let you select different features such as DHCP. Their values are documented at the top of the **TCP_CONFIG.LIB** library in the **LIB\Rabbit4000\TCPIP** directory. More information is available in the ***Dynamic C TCP/IP User's Manual***.

6.4.2 How to Set Up your Computer for Direct Connect

Follow these instructions to set up your PC or notebook. Check with your administrator if you are unable to change the settings as described here since you may need administrator privileges. The instructions are specifically for Windows 2000, but the interface is similar for other versions of Windows.

> **TIP:** If you are using a PC that is already on a network, you will disconnect the PC from that network to run these sample programs. Write down the existing settings before changing them to facilitate restoring them when you are finished with the sample programs and reconnect your PC to the network.

1. Go to the control panel (**Start > Settings > Control Panel**), and then double-click the Network icon.
2. Select the network interface card used for the Ethernet interface you intend to use (e.g., **TCP/IP Xircom Credit Card Network Adapter**) and click on the "Properties" button. Depending on which version of Windows your PC is running, you may have to select the "Local Area Connection" first, and then click on the "Properties" button to bring up the Ethernet interface dialog. Then "Configure" your interface card for a "10Base-T Half-Duplex" or an "Auto-Negotiation" connection on the "Advanced" tab.

   NOTE: Your network interface card will likely have a different name.

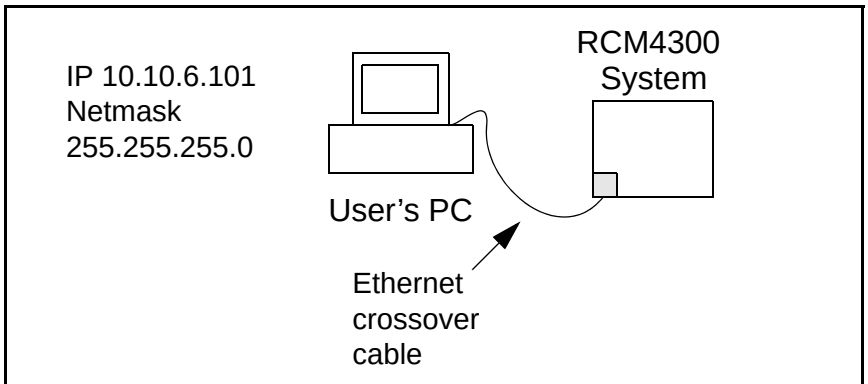
3. Now select the **IP Address** tab, and check **Specify an IP Address**, or select TCP/IP and click on "Properties" to assign an IP address to your computer (this will disable "obtain an IP address automatically"):

   IP Address : 10.10.6.101

   Netmask : 255.255.255.0

   Default gateway : 10.10.6.1

4. Click **<OK>** or **<Close>** to exit the various dialog boxes.



Direct Connection PC to RCM4300 Module

6.5 Run the PINGME.C Sample Program

Connect the crossover cable from your computer's Ethernet port to the RCM4300 module's RJ-45 Ethernet connector. Open this sample program from the **SAMPLES\TCPIP\ICMP** folder, compile the program, and start it running under Dynamic C. The crossover cable is connected from your computer's Ethernet adapter to the RCM4300 module's RJ-45 Ethernet connector. When the program starts running, the green **LINK** light on the RCM4300 module should be on to indicate an Ethernet connection is made. (Note: If the **LNK** light does not light, you may not be using a crossover cable, or if you are using a hub with straight-through cables perhaps the power is off on the hub.)

The next step is to ping the module from your PC. This can be done by bringing up the MS-DOS window and running the pingme program:

```
ping 10.10.6.101
```

or by **Start > Run**

and typing the entry

```
ping 10.10.6.101
```

The ping routine will ping the module four times and write a summary message on the screen describing the operation.

6.6 Running Additional Sample Programs With Direct Connect

The following sample programs are in the Dynamic C **SAMPLES\RCM4300\TCPIP** folder.

- **BROWSELED.C**—This program demonstrates a basic controller running a Web page. Two "device LEDs" are created along with two buttons to toggle them. Users can use their Web browser to change the state of the lights. The DS2 and DS3 LEDs on the Prototyping Board will match those on the Web page. As long as you have not modified the **TCPCONFIG 1** macro in the sample program, enter the following server address in your Web browser to bring up the Web page served by the sample program.

  http://10.10.6.100.

  Otherwise use the TCP/IP settings you entered in the **TCP_CONFIG.LIB** library.

- **PINGLED.C**—This program demonstrates ICMP by pinging a remote host. It will flash LEDs DS2 and DS3 on the Prototyping Board when a ping is sent and received.

- **SMTP.C**—This program demonstrates using the SMTP library to send an e-mail when the S2 and S3 switches on the Prototyping Board are pressed. LEDs DS2 and DS3 on the Prototyping Board will light up when e-mail is being sent.

6.7 Where Do I Go From Here?

NOTE: If you purchased your RCM4300 through a distributor or through a Rabbit Semiconductor partner, contact the distributor or partner first for technical support.

If there are any problems at this point:

- Use the Dynamic C **Help** menu to get further assistance with Dynamic C.
- Check the Rabbit Semiconductor Technical Bulletin Board and forums at www.rabbit.com/support/bb/ and at www.rabbit.com/forums/.
- Use the Technical Support e-mail form at www.rabbit.com/support/.

If the sample programs ran fine, you are now ready to go on.

Additional sample programs from the **SAMPLES\TCPIP** folder are described in the ***Dynamic C TCP/IP User's Manual***.

Please refer to the ***Dynamic C TCP/IP User's Manual*** to develop your own applications. ***An Introduction to TCP/IP*** provides background information on TCP/IP, and is available on the CD and on our Web site.

APPENDIX A. RCM4300 SPECIFICATIONS

Appendix A provides the specifications for the RCM4300, and describes the conformal coating.

A.1 Electrical and Mechanical Characteristics

Figure A-1 shows the mechanical dimensions for the RCM4300.

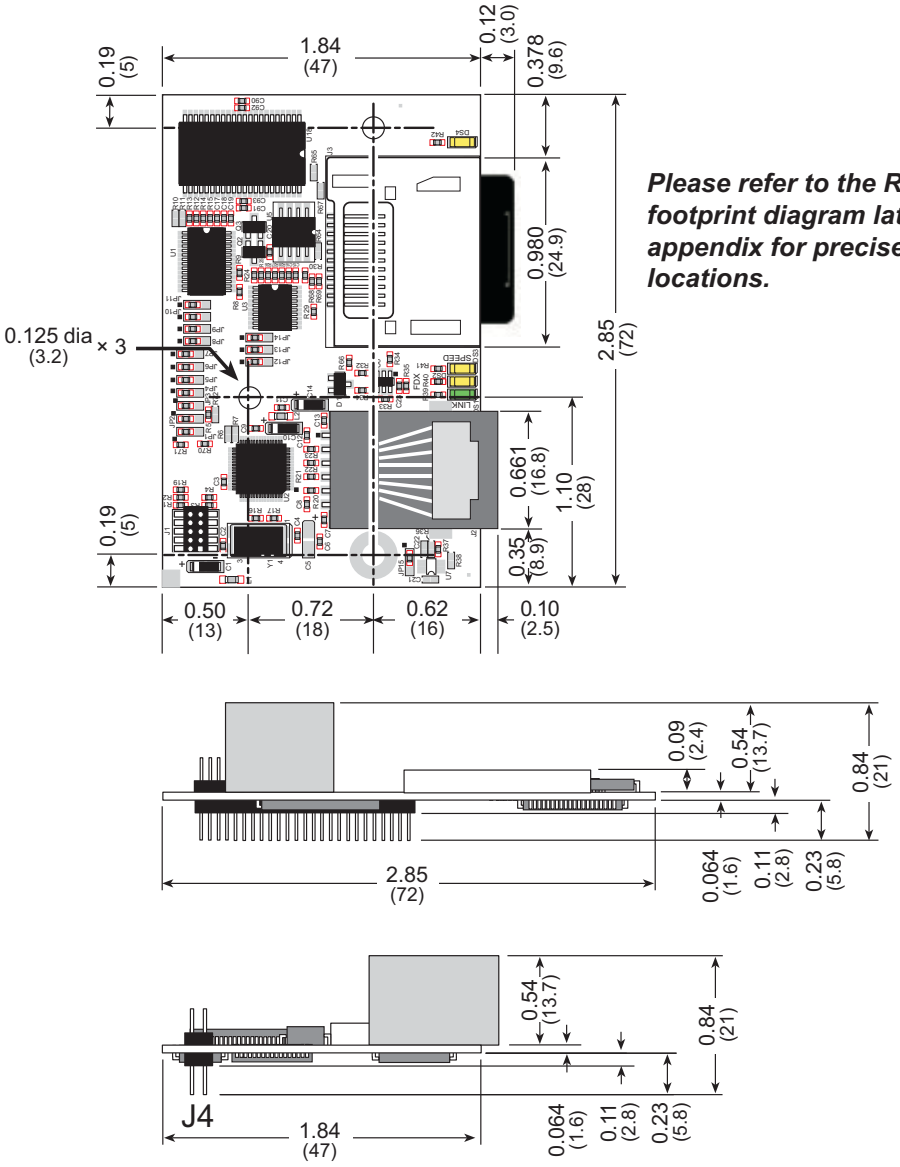


Figure A-1. RCM4300 Dimensions

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses. All dimensions have a manufacturing tolerance of ±0.01" (0.25 mm).

It is recommended that you allow for an “exclusion zone” of 0.04" (1 mm) around the RCM4300 in all directions when the RCM4300 is incorporated into an assembly that includes other printed circuit boards. An “exclusion zone” of 0.08" (2 mm) is recommended below the RCM4300 when the RCM4300 is plugged into another assembly. Figure A-2 shows this “exclusion zone.”

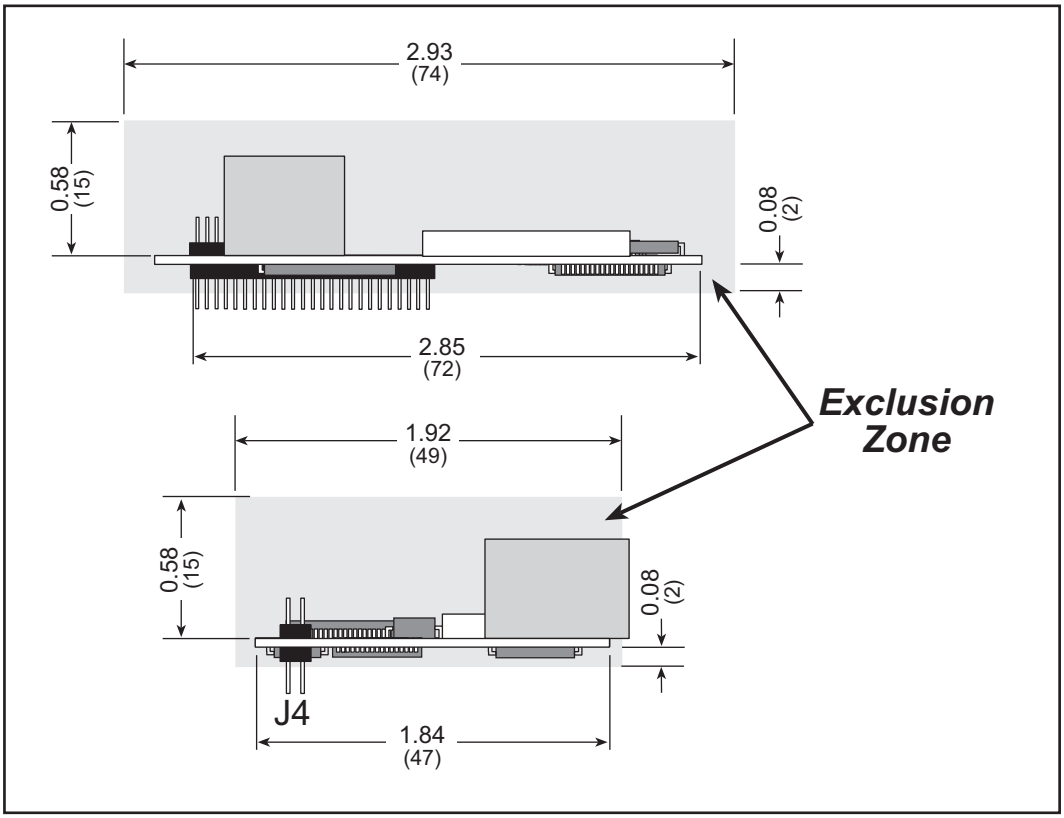


Figure A-2. RCM4300 “Exclusion Zone”

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses.

Table A-1 lists the electrical, mechanical, and environmental specifications for the RCM4300.

Table A-1. RabbitCore RCM4300 Specifications

Parameter	RCM4300	RCM4310
Microprocessor	Rabbit® 4000 at 58.98 MHz	
EMI Reduction	Spectrum spreader for reduced EMI (radiated emissions)	
Ethernet Port	10/100Base-T, RJ-45, 3 LEDs	
Data SRAM	512K (8-bit)	
Program Execution Fast SRAM	1MB (8-bit)	512K (8-bit)
Serial Flash Memory (program)	2MB	1MB
Memory (data storage)	*microSD™ Card* 128MB–1GB	*microSD™ Card* 128MB–1GB
LED Indicators	LINK/ACT (link/activity) FDX/COL (full-duplex/collisions) SPEED (on for 100Base-T Ethernet connection) SD (microSD™ mounted status)	
Backup Battery	Connection for user-supplied backup battery (to support RTC and data SRAM)	
General-Purpose I/O	28 parallel digital I/O lines: • configurable with four layers of alternate functions	36 parallel digital I/O lines: • configurable with four layers of alternate functions
Additional Inputs	2 startup mode, reset in, CONVERT	2 startup mode, reset in
Additional Outputs	Status, reset out, analog VREF	Status, reset out
Analog Inputs	8 channels single-ended or 4 channels differential Programmable gain 1, 2, 4, 5, 8, 10, 16, and 20 V/V	—
• A/D Converter Resolution	12 bits (11 bits single-ended)	
• A/D Conversion Time (including 120 µs raw count	180 µs	
Auxiliary I/O Bus	Can be configured for 8 data lines and 5 address lines (shared with parallel I/O lines), plus I/O read/write	

Table A-1. RabbitCore RCM4300 Specifications (continued)

Parameter	RCM4300	RCM4310
Serial Ports	5 shared high-speed, CMOS-compatible ports: • all 5 configurable as asynchronous (with IrDA), 4 as clocked serial (SPI), and 1 as SDLC/HDLC • 1 clocked serial port shared with programming port • 1 clocked serial port shared with A/D converter, serial flash, and microSD™ card	6 shared high-speed, CMOS-compatible ports: • all 6 configurable as asynchronous (with IrDA), 4 as clocked serial (SPI), and 2 as SDLC/HDLC • 1 clocked serial port shared with programming port • 1 clocked serial port shared with serial flash and microSD™ card
Serial Rate	Maximum asynchronous baud rate = CLK/8	
Slave Interface	Slave port allows the RCM4300 to be used as an intelligent peripheral device slaved to a master processor	
Real-Time Clock	Yes	
Timers	Ten 8-bit timers (6 cascadable from the first), one 10-bit timer with 2 match registers, and one 16-bit timer with 4 outputs and 8 set/reset registers	
Watchdog/Supervisor	Yes	
Pulse-Width Modulators	4 PWM registers with 10-bit free-running counter and priority interrupts	
Input Capture	2-channel input capture can be used to time input signals from various port pins	
Quadrature Decoder	2-channel quadrature decoder accepts inputs from external incremental encoder modules	
Power (pins unloaded)	3.0–3.6 V DC, 350 mA (typ.) @ 3.3 V, 385 mA @ 3.6 V and 85°C (max.)	
Operating Temperature	-20°C to +85°C	
Humidity	5% to 95%, noncondensing	
Connectors	One 2 × 25, 1.27 mm pitch IDC signal header One microSD™ Card socket One 2 × 5, 1.27 mm pitch IDC programming header	
Board Size	1.84" × 2.85" × 0.84" (47 mm × 72 mm × 21 mm)	

A.1.1 A/D Converter

Table A-2 shows some of the important A/D converter specifications. For more details, refer to the ADS7870 data sheet.

Table A-2. A/D Converter Specifications

Parameter	Test Conditions	Typ	Max
Analog Input Characteristics			
Input Capacitance		4 – 9.7 pF	
Input Impedance			
Common-Mode		6 MΩ	
Differential Mode		7 MΩ	
Static Accuracy			
Resolution			
Single-Ended Mode		11 bits	
Differential Mode		12 bits	
Integral Linearity		±2 LSB	±2.5 LSB
Differential Linearity		±0.5 LSB	
Dynamic Characteristics			
Throughput Rate		52 ksamples/s	
Voltage Reference			
Accuracy	V_{ref} = 2.048 V and 2.5 V	±0.05%	±0.25%
Buffer Amp Source Current		20 mA	
Buffer Amp Sink Current		200 µA	
Short-Circuit Current		20 mA	

A.1.2 Headers

The RCM4300 uses headers at J1 and J4 for physical connection to other boards. J4 is a 2 × 25 SMT header with a 1.27 mm pin spacing. J1, the programming port, is a 2 × 5 header with a 1.27 mm pin spacing.

Figure A-3 shows the layout of another board for the RCM4300 to be plugged into. These reference design values are relative to the mounting hole.

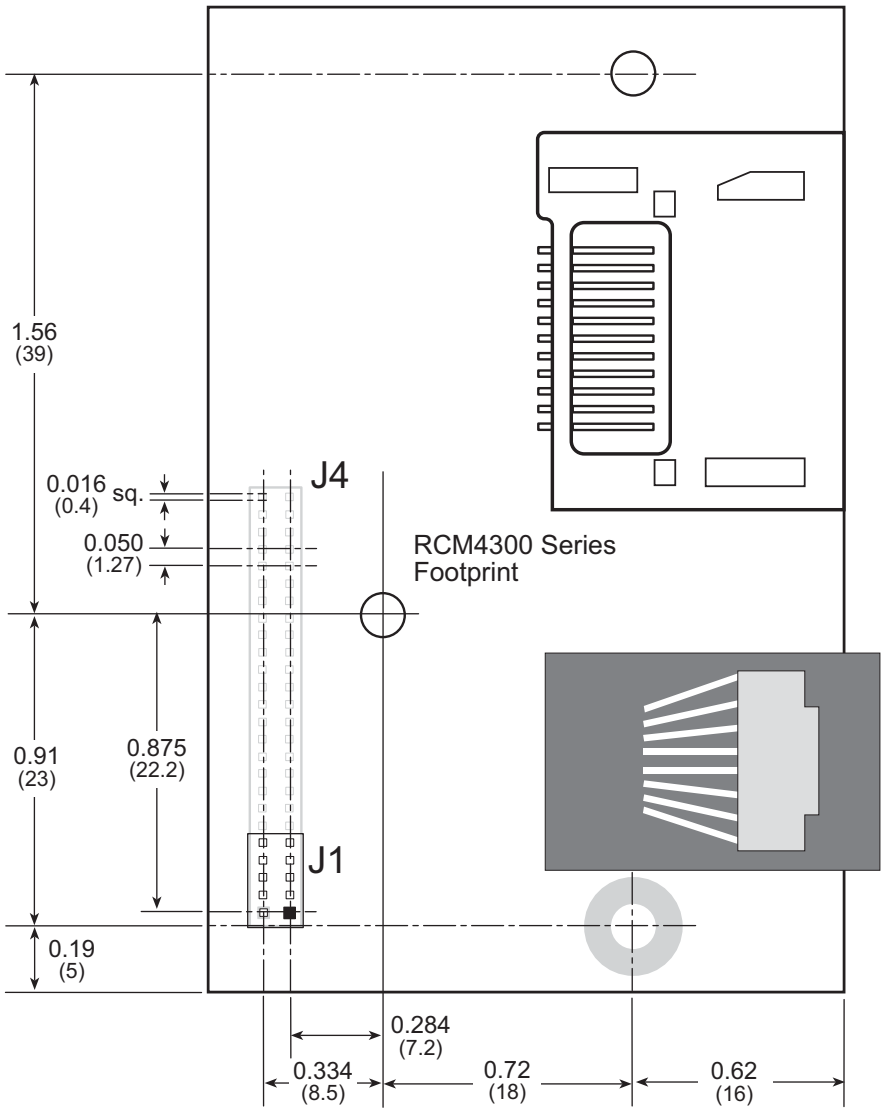


Figure A-3. User Board Footprint for RCM4300

A.2 Rabbit 4000 DC Characteristics

Table A-3. Rabbit 4000 Absolute Maximum Ratings

Symbol	Parameter	Maximum Rating
T_A	Operating Temperature	-40° to +85°C
T_S	Storage Temperature	-55° to +125°C
V_{IH}	Maximum Input Voltage	VDD_{IO} + 0.3 V (max. 3.6 V)
VDD_{IO}	Maximum Operating Voltage	3.6 V

Stresses beyond those listed in Table A-3 may cause permanent damage. The ratings are stress ratings only, and functional operation of the Rabbit 4000 chip at these or any other conditions beyond those indicated in this section is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect the reliability of the Rabbit 4000 chip.

Table A-4 outlines the DC characteristics for the Rabbit 4000 at 3.3 V over the recommended operating temperature range from T_A = –40°C to +85°C, VDD_{IO} = 3.0 V to 3.6 V.

Table A-4. 3.3 Volt DC Characteristics

Symbol	Parameter	Min	Typ	Max
VDD_{IO}	I/O Ring Supply Voltage, 3.3 V	3.0 V	3.3 V	3.6 V
	I/O Ring Supply Voltage, 1.8 V	1.65 V	1.8 V	1.90 V
V_{IH}	High-Level Input Voltage (VDD_{IO} = 3.3 V)		2.0 V	
V_{IL}	Low-Level Input Voltage (VDD_{IO} = 3.3 V)		0.8 V	
V_{OH}	High-Level Output Voltage (VDD_{IO} = 3.3 V)		2.4 V	
V_{OL}	Low-Level Output Voltage (VDD_{IO} = 3.3 V)		0.4 V	
I_{IO}	I/O Ring Current @ 29.4912 MHz, 3.3 V, 25°C			12.2 mA
I_{DRIVE}	All other I/O (except TXD+, TXDD+, TXD-, TXDD-)			8 mA

A.3 I/O Buffer Sourcing and Sinking Limit

Unless otherwise specified, the Rabbit I/O buffers are capable of sourcing and sinking 8 mA of current per pin at full AC switching speed. Full AC switching assumes a 29.4 MHz CPU clock with the clock doubler enabled and capacitive loading on address and data lines of less than 70 pF per pin. The absolute maximum operating voltage on all I/O is 3.6 V.

A.4 Bus Loading

You must pay careful attention to bus loading when designing an interface to the RCM4300. This section provides bus loading information for external devices.

Table A-5 lists the capacitance for the various RCM4300 I/O ports.

Table A-5. Capacitance of Rabbit 4000 I/O Ports

I/O Ports	Input Capacitance (pF)	Output Capacitance (pF)
Parallel Ports A to E	12	14

Table A-6 lists the external capacitive bus loading for the various RCM4300 output ports. Be sure to add the loads for the devices you are using in your custom system and verify that they do not exceed the values in Table A-6.

Table A-6. External Capacitive Bus Loading -40°C to +85°C

Output Port	Clock Speed (MHz)	Maximum External Capacitive Loading (pF)
All I/O lines with clock doubler enabled	58.98	100

Figure A-4 shows a typical timing diagram for the Rabbit 4000 microprocessor external I/O read and write cycles.

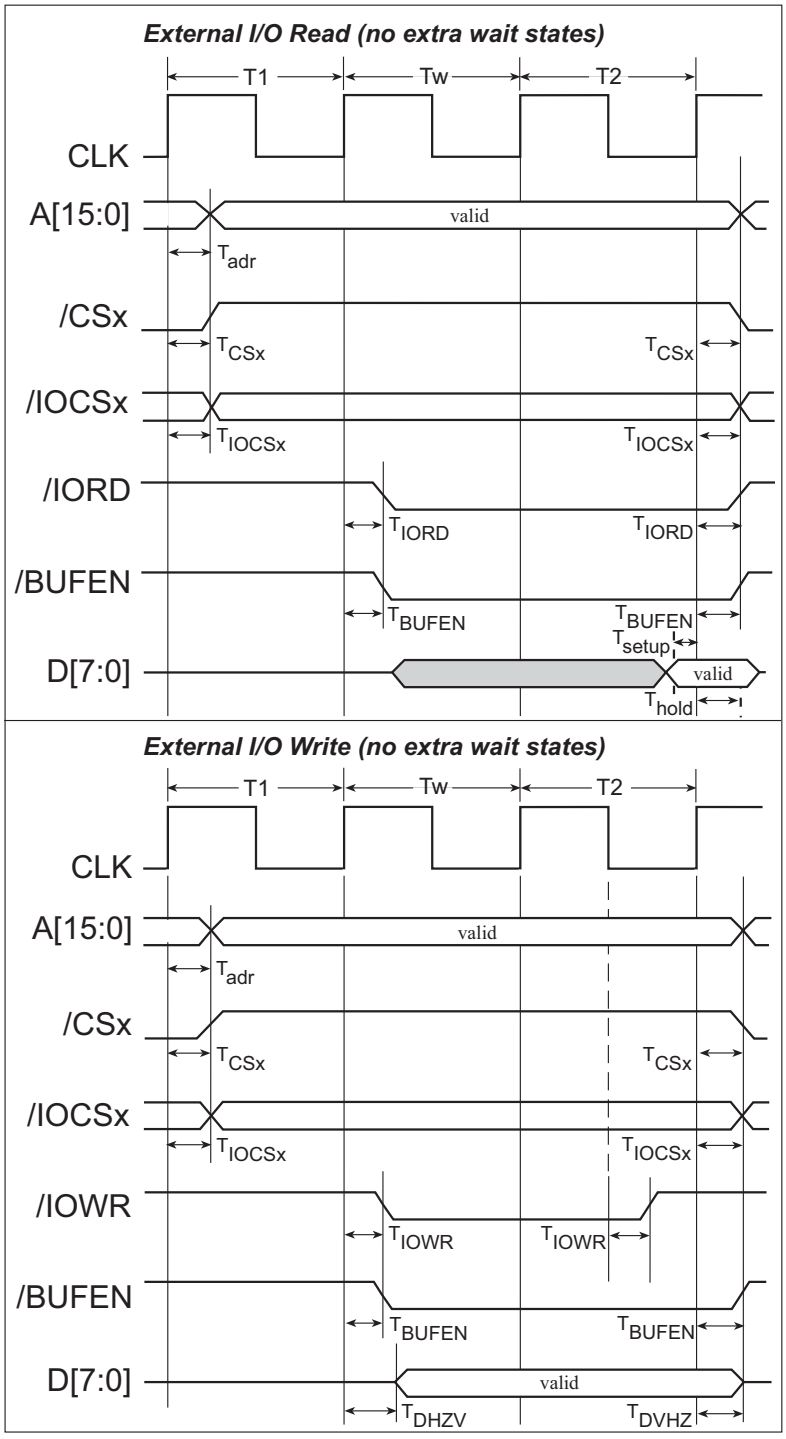


Figure A-4. I/O Read and Write Cycles—No Extra Wait States

NOTE: /IOCSx can be programmed to be active low (default) or active high.

Table A-7 lists the delays in gross memory access time for several values of VDD_{IO}.

Table A-7. Preliminary Data and Clock Delays

VDD_{IO} (V)	Clock to Address Output Delay (ns)			Data Setup Time Delay (ns)	Worst-Case Spectrum Spreader Delay (ns)		
	30 pF	60 pF	90 pF		0.5 ns setting no dbl / dbl	1 ns setting no dbl / dbl	2 ns setting no dbl / dbl
3.3	6	8	11	1	2.3 / 2.3	3 / 4.5	4.5 / 9
1.8	18	24	33	3	7 / 6.5	8 / 12	11 / 22

The measurements are taken at the 50% points under the following conditions.

- T = -40°C to 85°C, V = VDD_{IO} ±10%
- Internal clock to nonloaded CLK pin delay ≤ 1 ns @ 85°C/3.0 V

The clock to address output delays are similar, and apply to the following delays.

- T_{adr}, the clock to address delay
- T_{CSx}, the clock to memory chip select delay
- T_{IOCSx}, the clock to I/O chip select delay
- T_{IORD}, the clock to I/O read strobe delay
- T_{IOWR}, the clock to I/O write strobe delay
- T_{BUFEN}, the clock to I/O buffer enable delay

The data setup time delays are similar for both T_{setup} and T_{hold}.

When the spectrum spreader is enabled with the clock doubler, every other clock cycle is shortened (sometimes lengthened) by a maximum amount given in the table above. The shortening takes place by shortening the high part of the clock. If the doubler is not enabled, then every clock is shortened during the low part of the clock period. The maximum shortening for a pair of clocks combined is shown in the table.

Rabbit Semiconductor's Technical Note TN227, ***Interfacing External I/O with Rabbit 2000/3000 Designs***, contains suggestions for interfacing I/O devices to the Rabbit 3000 and Rabbit 4000 microprocessors.

A.5 Jumper Configurations

Figure A-5 shows the jumper locations used to configure the various RCM4300 options. The black square indicates pin 1.

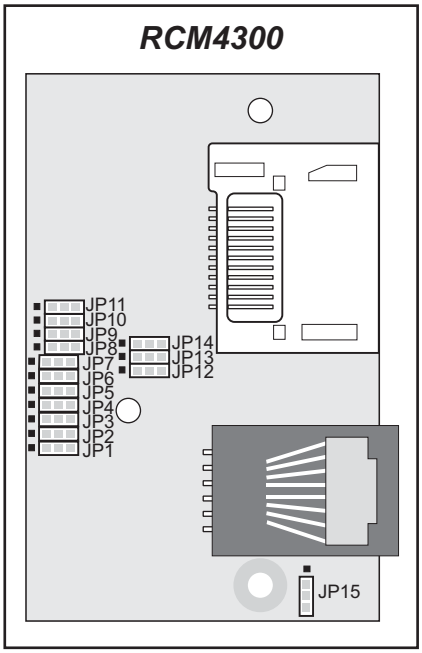


Figure A-5. Location of RCM4300 Configurable Positions

Table A-8 lists the configuration options.

Table A-8. RCM4300 Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	PE6 or SMODE1 Output on J4 pin 38	1–2	PE6	×
		2–3	SMODE1	
JP2	PE5 or SMODE0 Output on J4 pin 37	1–2	PE5	×
		2–3	SMODE0	
JP3	PE7 or STATUS Output on J4 pin 39	1–2	PE7	×
		2–3	STATUS	
JP4	LN0 or PD0 on J4 pin 40	1–2	LN0	RCM4300
		2–3	PD0	RCM4310
JP5	LN1 or PD1 on J4 pin 41	1–2	LN1	RCM4300
		2–3	PD1	RCM4310

Table A-8. RCM4300 Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP6	LN2 or PD2 on J4 pin 42	1–2	LN2	RCM4300
		2–3	PD2	RCM4310
JP7	LN3 or PD3 on J4 pin 43	1–2	LN3	RCM4300
		2–3	PD3	RCM4310
JP8	LN4 or PD4 on J4 pin 44	1–2	LN4	RCM4300
		2–3	PD4	RCM4310
JP9	LN6 or PD6 on J4 pin 46	1–2	LN6	RCM4300
		2–3	PD6	RCM4310
JP10	LN5 or PD5 on J4 pin 45	1–2	LN5	RCM4300
		2–3	PD5	RCM4310
JP11	LN7 or PD7 on J4 pin 47	1–2	LN7	RCM4300
		2–3	PD7	RCM4310
JP12	PC4_TxB or PC2_TxC to A/D converter SDI	1–2	PC4_TxB	RCM4300
		2–3	PC2_TxC	
JP13	PC5_RxB or PC3_RxC to A/D converter BUSY	1–2	PC5_RxB	RCM4300
		2–3	PC3_RxC	
JP14	PB0 or PD2 to A/D converter SCLK	1–2	PB0	RCM4300
		2–3	PD2	
JP15	LED DS2 Display	1–2	FDX/COL displayed by LED DS2	✕
		2–3	optional ACT displayed by LED DS2	

NOTE: The jumper connections are made using 0 Ω surface-mounted resistors.

A.6 Conformal Coating

The areas around the 32 kHz real-time clock crystal oscillator have had the Dow Corning silicone-based 1-2620 conformal coating applied. The conformally coated area is shown in Figure A-6. The conformal coating protects these high-impedance circuits from the effects of moisture and contaminants over time.

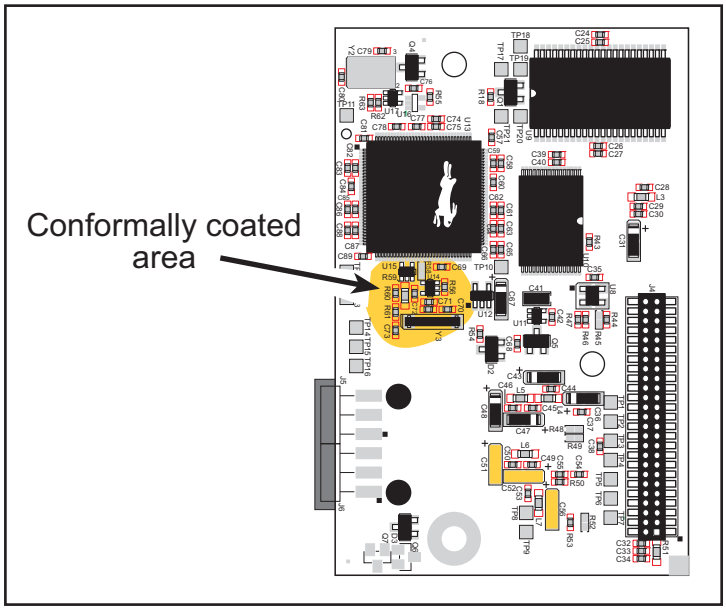


Figure A-6. RCM4300 Areas Receiving Conformal Coating

Any components in the conformally coated area may be replaced using standard soldering procedures for surface-mounted components. A new conformal coating should then be applied to offer continuing protection against the effects of moisture and contaminants.

NOTE: For more information on conformal coatings, refer to Rabbit Semiconductor's Technical Note 303, ***Conformal Coatings***.

APPENDIX B. PROTOTYPING BOARD

Appendix B describes the features and accessories of the Prototyping Board, and explains the use of the Prototyping Board to demonstrate the RCM4300 and to build prototypes of your own circuits. The Prototyping Board has power-supply connections and also provides some basic I/O peripherals (RS-232, LEDs, and switches), as well as a prototyping area for more advanced hardware development.

B.1 Introduction

The Prototyping Board included in the Development Kit makes it easy to connect an RCM4300 module to a power supply and a PC workstation for development. It also provides some basic I/O peripherals (RS-232, LEDs, and switches), as well as a prototyping area for more advanced hardware development.

For the most basic level of evaluation and development, the Prototyping Board can be used without modification.

As you progress to more sophisticated experimentation and hardware development, modifications and additions can be made to the board without modifying the RCM4300 module.

The Prototyping Board is shown below in Figure B-1, with its main features identified.

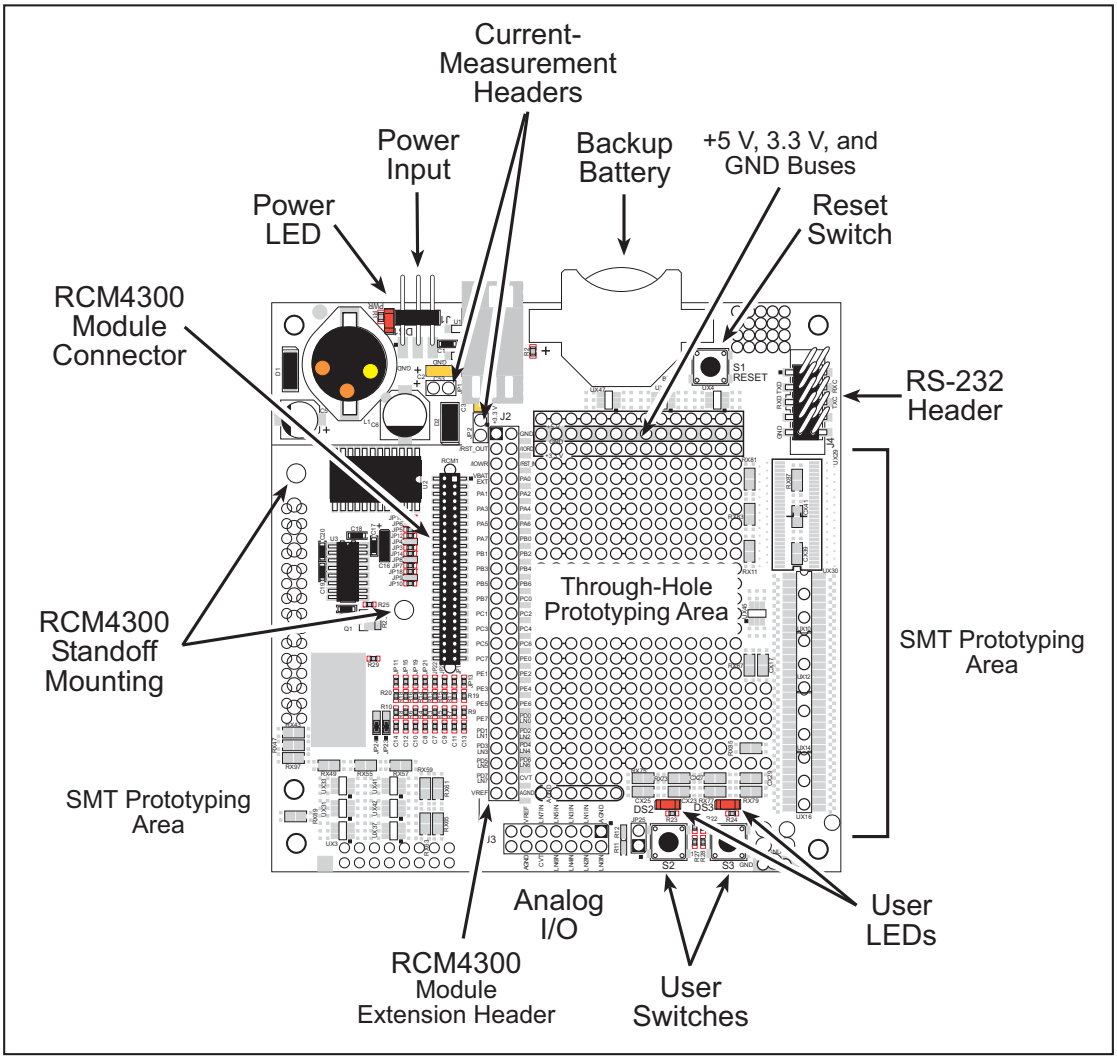


Figure B-1. Prototyping Board

B.1.1 Prototyping Board Features

- **Power Connection**—A a 3-pin header is provided for connection to the power supply. Note that the 3-pin header is symmetrical, with both outer pins connected to ground and the center pin connected to the raw V+ input. The cable of the AC adapter provided with the North American version of the Development Kit is terminated with a header plug that connects to the 3-pin header in either orientation. The header plug leading to bare leads provided for overseas customers can be connected to the 3-pin header in either orientation.

  Users providing their own power supply should ensure that it delivers 8–24 V DC at 8 W. The voltage regulators will get warm while in use.

- **Regulated Power Supply**—The raw DC voltage provided at the 3-pin header is routed to a 5 V switching voltage regulator, then to a separate 3.3 V linear regulator. The regulators provide stable power to the RCM4300 module and the Prototyping Board.

- **Power LED**—The power LED lights whenever power is connected to the Prototyping Board.

- **Reset Switch**—A momentary-contact, normally open switch is connected directly to the RCM4300's **/RESET_IN** pin. Pressing the switch forces a hardware reset of the system.

- **I/O Switches and LEDs**—Two momentary-contact, normally open switches are connected to the PB4 and PB5 pins of the RCM4300 module and may be read as inputs by sample applications.

  Two LEDs are connected to the PB2 and PB3 pins of the RCM4300 module, and may be driven as output indicators by sample applications.

- **Prototyping Area**—A generous prototyping area has been provided for the installation of through-hole components. +3.3 V, +5 V, and Ground buses run around the edge of this area. Several areas for surface-mount devices are also available. (Note that there are SMT device pads on both top and bottom of the Prototyping Board.) Each SMT pad is connected to a hole designed to accept a 30 AWG solid wire.

- **Module Extension Header**—The complete pin set of the RCM4300 module is duplicated at header J2. Developers can solder wires directly into the appropriate holes, or, for more flexible development, a 2 × 25 header strip with a 0.1" pitch can be soldered into place. See Figure B-4 for the header pinouts.

  NOTE: The same Prototyping Board can be used for several series of RabbitCore modules, and so the signals at J2 depend on the signals available on the specific RabbitCore module.

- **Analog Inputs Header**—The Prototyping Board's analog signals are presented at header J3. These analog signals are connected via attenuator/filter circuits on the Prototyping Board to the corresponding analog inputs on the RCM4300 module. Developers can solder wires directly into the appropriate holes, or, for more flexible development, a 2 × 7 header strip with a 0.1" pitch can be soldered into place. See Figure B-4 for the header pinouts.

- **RS-232**—Two 3-wire or one 5-wire RS-232 serial ports are available on the Prototyping Board at header J4. A 10-pin 0.1" pitch header strip installed at J4 allows you to connect a ribbon cable that leads to a standard DE-9 serial connector.
- **Current Measurement Option**—You may cut the trace below header JP1 on the bottom side of the Prototyping Board and install a 1 × 2 header strip from the Development Kit to allow you to use an ammeter across the pins to measure the current drawn from the +5 V supply. Similarly, you may cut the trace below header JP2 on the bottom side of the Prototyping Board and install a 1 × 2 header strip from the Development Kit to allow you to use an ammeter across the pins to measure the current drawn from the +3.3 V supply.
- **Backup Battery**—A 2032 lithium-ion battery rated at 3.0 V, 220 mA·h, provides battery backup for the RCM4300 SRAM and real-time clock.

B.2 Mechanical Dimensions and Layout

Figure B-2 shows the mechanical dimensions and layout for the Prototyping Board.

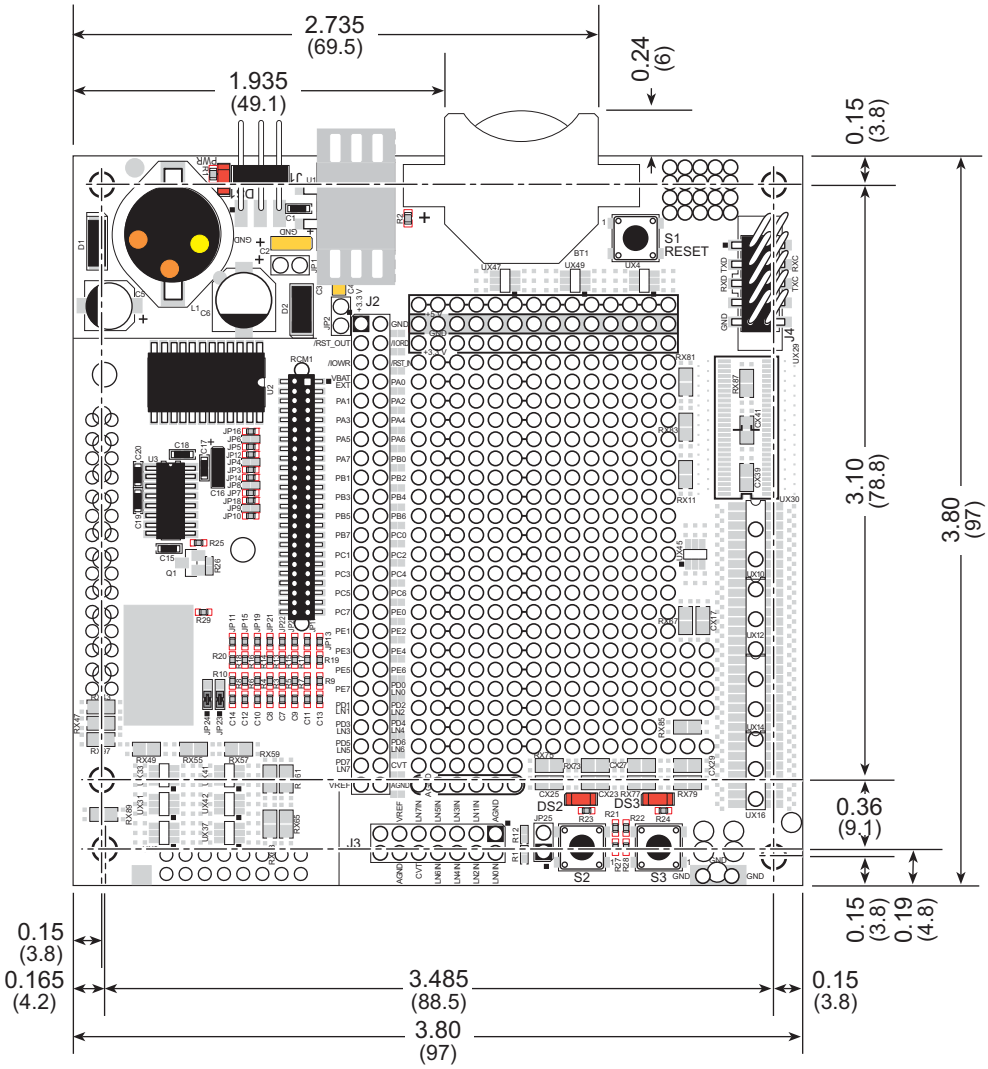


Figure B-2. Prototyping Board Dimensions

Table B-1 lists the electrical, mechanical, and environmental specifications for the Prototyping Board.

Table B-1. Prototyping Board Specifications

Parameter	Specification
Board Size	3.80" × 3.80" × 0.48" (97 mm × 97 mm × 12 mm)
Operating Temperature	0°C to +70°C
Humidity	5% to 95%, noncondensing
Input Voltage	8 V to 24 V DC
Maximum Current Draw (including user-added circuits)	800 mA max. for +3.3 V supply, 1 A total +3.3 V and +5 V combined
Prototyping Area	1.3" × 2.0" (33 mm × 50 mm) throughhole, 0.1" spacing, additional space for SMT components
Connectors	One 2 × 25 header socket, 1.27 mm pitch, to accept RCM4300 One 1 × 3 IDC header for power-supply connection One 2 × 5 IDC RS-232 header, 0.1" pitch Two unstuffed header locations for analog and RCM4300 signals 25 unstuffed 2-pin header locations for optional configurations

B.3 Power Supply

The RCM4300 requires a regulated 3.0 V – 3.6 V DC power source to operate. Depending on the amount of current required by the application, different regulators can be used to supply this voltage.

The Prototyping Board has an onboard +5 V switching power regulator from which a +3.3 V linear regulator draws its supply. Thus both +5 V and +3.3 V are available on the Prototyping Board.

The Prototyping Board itself is protected against reverse polarity by a Shottky diode at D2 as shown in Figure B-3.

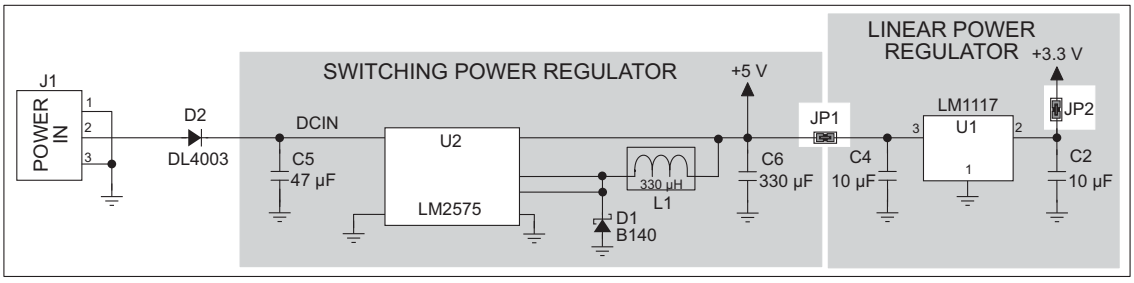


Figure B-3. Prototyping Board Power Supply

B.4 Using the Prototyping Board

The Prototyping Board is actually both a demonstration board and a prototyping board. As a demonstration board, it can be used to demonstrate the functionality of the RCM4300 right out of the box without any modifications to either board.

The Prototyping Board comes with the basic components necessary to demonstrate the operation of the RCM4300. Two LEDs (DS2 and DS3) are connected to PB2 and PB3, and two switches (S2 and S3) are connected to PB4 and PB5 to demonstrate the interface to the Rabbit 4000 microprocessor. Reset switch S1 is the hardware reset for the RCM4300.

The Prototyping Board provides the user with RCM4300 connection points brought out conveniently to labeled points at header J2 on the Prototyping Board. Although header J2 is unstuffed, a 2 × 25 header is included in the bag of parts. RS-232 signals (Serial Ports C and D) are available on header J4. A header strip at J4 allows you to connect a ribbon cable, and a ribbon cable to DB9 connector is included with the Development Kit. The pinouts for these locations are shown in Figure B-4.

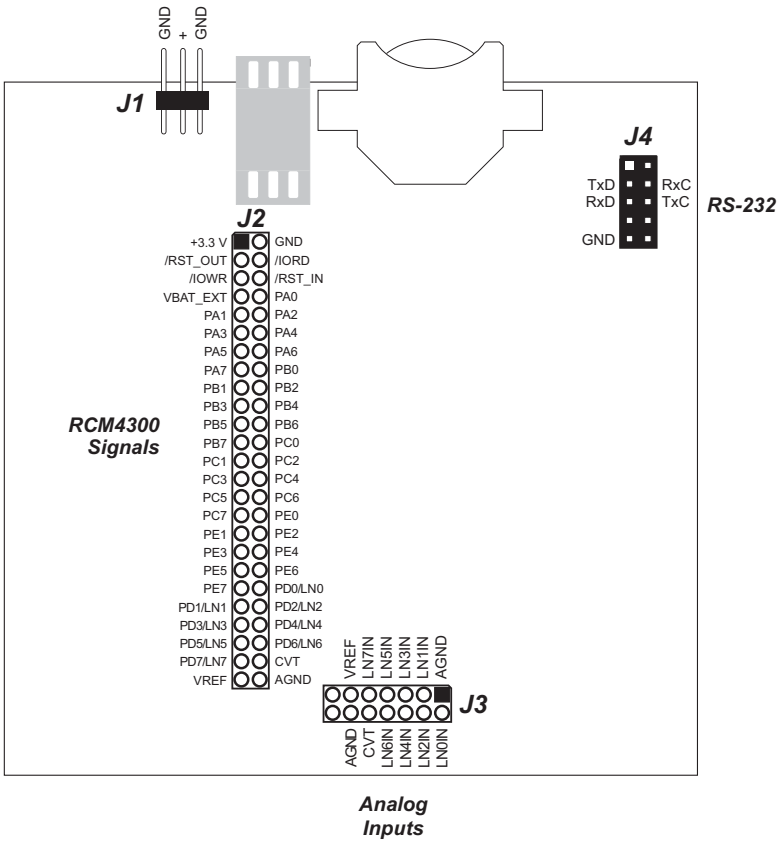


Figure B-4. Prototyping Board Pinout

The analog signals are brought out to labeled points at header location J3 on the Prototyping Board. Although header J3 is unstuffed, a 2 × 7 header can be added. Note that analog signals are only available from the RCM4300 included in the Development Kit — the RCM4210 model does not have an A/D converter.

All signals from the RCM4300 module are available on header J2 of the Prototyping Board. The remaining ports on the Rabbit 4000 microprocessor are used for RS-232 serial communication. Table B-2 lists the signals on header J2 and explains how they are used on the Prototyping Board.

Table B-2. Use of RCM4300 Signals on the Prototyping Board

Pin	Pin Name	Prototyping Board Use
1	+3.3 V	+3.3 V power supply
2	GND	
3	/RST_OUT	Reset output from reset generator
4	/IORD	External read strobe
5	/IOWR	External write strobe
6	/RESET_IN	Input to reset generator
8–15	PA0–PA7	Output, pulled low
16	PB0	CLKB (shared by serial flash, microSD™ Card, and A/D converter if equipped)
17	PB1	Programming port CLKA
18	PB2	LED DS2 (normally high/off)
19	PB3	LED DS3 (normally high/off)
20	PB4	Switch S2 (normally open/pulled up)
21	PB5	Switch S3 (normally open/pulled up)
22–23	PB6–PB7	Output, pulled high
24–25	PC0–PC1	Serial Port D (RS-232, header J4) (high)
26–27	PC2–PC3	Serial Port C (high)
28–29	PC4–PC5	Serial Port B (shared by serial flash, microSD™ Card, and A/D converter if equipped)
30–31	PC6–PC7	Serial Port A (programming port) (high)
32–33	PE0–PE1	Output (high)
34	PE2	Not brought out to Prototyping Board
35–39	PE3–PE7	Output (high)
40–47	LN0–LN7	A/D converter inputs (RCM4300 only)*
48	CONVERT	A/D converter CONVERT input (RCM4300 only)*
49	VREF	A/D converter reference voltage (RCM4300 only)*
50	AGND	A/D converter ground (RCM4300 only)*

* PD0–PD7 (output, high) are available on these pins for the RCM4310.

There is a 1.3" × 2" through-hole prototyping space available on the Prototyping Board. The holes in the prototyping area are spaced at 0.1" (2.5 mm). +3.3 V, +5 V, and GND traces run along the top edge of the prototyping area for easy access. Small to medium circuits can be prototyped using point-to-point wiring with 20 to 30 AWG wire between the prototyping area, the +3.3 V, +5 V, and GND traces, and the surrounding area where surface-mount components may be installed. Small holes are provided around the surface-mounted components that may be installed around the prototyping area.

B.4.1 Adding Other Components

There are pads for 28-pin TSSOP devices, 16-pin SOIC devices, and 6-pin SOT devices that can be used for surface-mount prototyping with these devices. There are also pads that can be used for SMT resistors and capacitors in an 0805 SMT package. Each component has every one of its pin pads connected to a hole in which a 30 AWG wire can be soldered (standard wire wrap wire can be soldered in for point-to-point wiring on the Prototyping Board). Because the traces are very thin, carefully determine which set of holes is connected to which surface-mount pad.

B.4.2 Measuring Current Draw

The Prototyping Board has a current-measurement feature available at header locations JP1 and JP2 for the +5 V and +3.3 V supplies respectively. To measure current, you will have to cut the trace on the bottom side of the Prototyping Board corresponding to the power supply or power supplies whose current draw you will be measuring. Header locations JP1 and JP2 are shown in Figure B-5. Then install a 1 × 2 header strip from the Development Kit on the top side of the Prototyping Board at the header location(s) whose trace(s) you cut. The header strip(s) will allow you to use an ammeter across their pins to measure the current drawn from that supply. Once you are done measuring the current, place a jumper across the header pins to resume normal operation.

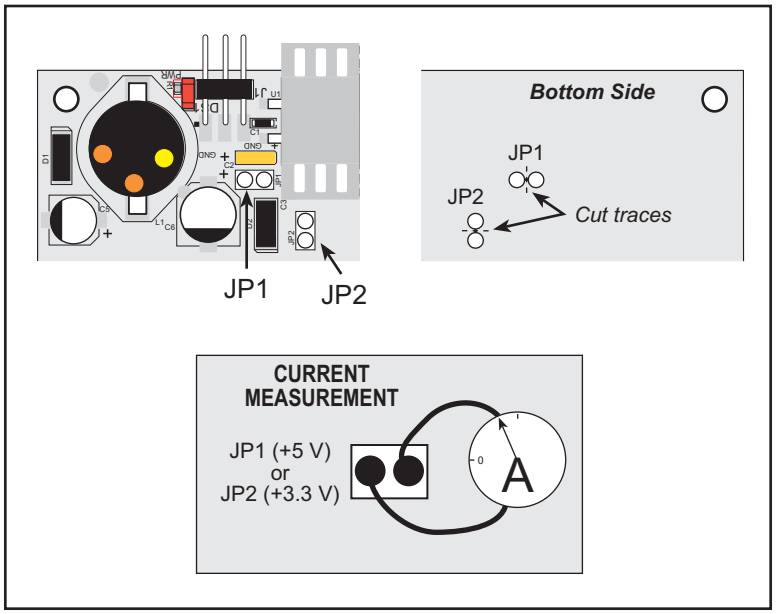


Figure B-5. Prototyping Board Current-Measurement Option

NOTE: Once you have cut the trace below header location JP1 or JP2, you must either be using the ammeter or have a jumper in place in order for power to be delivered to the Prototyping Board.

B.4.3 Analog Features (RCM4300 only)

The Prototyping Board has typical support circuitry installed to complement the ADS7870 A/D converter on the RCM4300 model (the A/D converter is not available on the RCM4210 model).

B.4.3.1 A/D Converter Inputs

Figure B-6 shows a pair of A/D converter input circuits. The resistors form an approx. 11:1 attenuator, and the capacitor filters noise pulses from the A/D converter input. The 470 Ω inline jumpers allow other configurations (see Table B-6) and provide digital isolation when you are not using an A/D converter (Parallel Port D is available). These jumpers optimize using RabbitCore modules with or without A/D converters—if you are designing your own circuit, the best performance for the A/D converter would be realized with 0 Ω resistors.

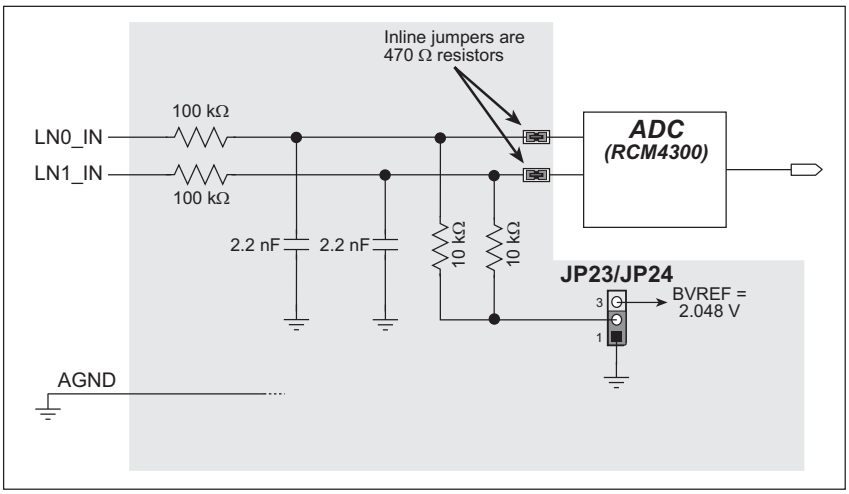


Figure B-6. A/D Converter Inputs

The A/D converter chip can make either single-ended or differential measurements depending on the value of the **opmode** parameter in the software function call. Adjacent A/D converter inputs are paired to make differential measurements. The default setup on the Prototyping Board is to measure only positive voltages for the ranges listed in Table B-3.

Table B-3. Positive A/D Converter Input Voltage Ranges

Min. Voltage (V)	Max. Voltage (with prescaler) (V)	Gain Multiplier	A/D Converter Actual Gain	Resolution (mV)
0.0	+22.528	×1	1	11
0.0	+11.264	×2	1.8	5.5
0.0	+5.632	×4	3.6	2.75
0.0	+4.506	×5	4.5	2.20
0.0	+2.816	×8	7.2	1.375
0.0	+2.253	×10	9.0	1.100
0.0	+1.408	×16	14.4	0.688
0.0	+1.126	×20	18	0.550

Many other possible ranges are possible by physically changing the resistor values that make up the attenuator circuit.

> **NOTE:** Analog input LN7_IN does not have the 10 kΩ resistor installed, and so no resistor attenuator is available, limiting its maximum input voltage to 2 V. This input is intended to be used for a thermistor that you may install at header location JP25.

It is also possible to read a negative voltage on LN0_IN–LN5_IN by moving the 0 Ω jumper (see Figure B-6) on header JP23 or JP24 associated with the A/D converter input from analog ground to the reference voltage generated and buffered by the A/D converter. Adjacent input channels are paired; moving the jumper on JP 23 changes both of the paired channels (LN4_IN–LN5_IN), and moving the jumper on JP24 changes LN0_IN–LN1_IN and LN2_IN–LN3_IN. At the present time Rabbit Semiconductor does not offer the software drivers to work with single-ended negative voltages, but the differential mode described below may be used to measure negative voltages.

Differential measurements require two channels. As the name *differential* implies, the difference in voltage between the two adjacent channels is measured rather than the difference between the input and analog ground. Voltage measurements taken in differential mode have a resolution of 12 bits, with the 12th bit indicating whether the difference is positive or negative.

The A/D converter chip can only accept positive voltages, as explained in Section 4.4. Both differential inputs must be referenced to analog ground, and ***both inputs must be positive with respect to analog ground***. Table B-4 provides the differential voltage ranges for this setup.

Table B-4. Differential Voltage Ranges

Min. Differential Voltage (V)	Max. Differential Voltage (with prescaler) (V)	Gain Multiplier	A/D Converter Actual Gain	Resolution (mV)
0	±22.528	×1	1	11
0	±11.264	×2	1.8	5.5
0	±5.632	×4	3.6	2.75
0	±4.506	×5	4.5	2.20
0	±2.816	×8	7.2	1.375
0	±2.253	×10	9.0	1.100
0	±1.408	×16	14.4	0.688
0	±1.126	×20	18	0.550

B.4.3.2 Thermistor Input

Analog input LN7_IN on the Prototyping Board was designed specifically for use with a thermistor at JP25 in conjunction with the **THERMISTOR.C** sample program, which demonstrates how to use the analog input to measure temperature, which will be displayed in the Dynamic C **STDIO** window. The sample program is targeted specifically for the thermistor included with the Development Kit with R_0 @ 25°C = 3 kΩ and β 25/85 = 3965. Be sure to use the applicable R_0 and β values for your thermistor if you use another thermistor.

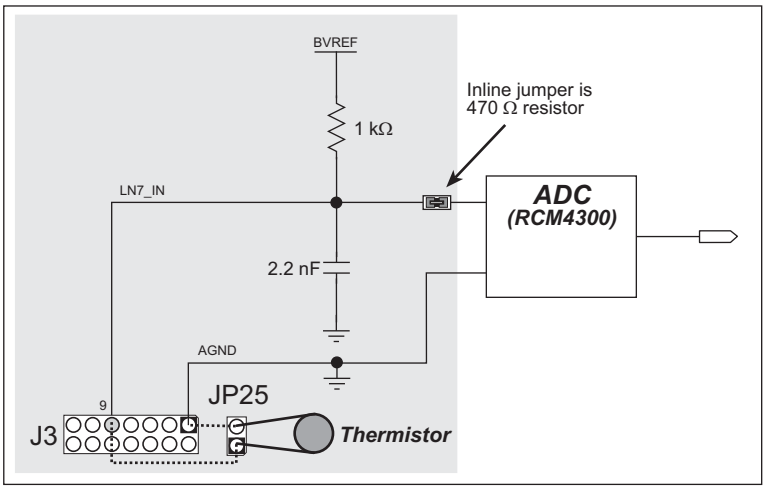


Figure B-7. Prototyping Board Thermistor Input

B.4.3.3 A/D Converter Calibration

To get the best results form the A/D converter, it is necessary to calibrate each mode (single-ended or differential) for each of its gains. It is imperative that you calibrate each of the A/D converter inputs in the same manner as they are to be used in the application. For example, if you will be performing floating differential measurements or differential measurements using a common analog ground, then calibrate the A/D converter in the corresponding manner. The calibration must be done with the JP23/JP24 selection jumpers in the desired position (see Figure B-6). If a calibration is performed and a jumper is subsequently moved, the corresponding input(s) must be recalibrated. The calibration table in software only holds calibration constants based on mode, channel, and gain. ***Other factors affecting the calibration must be taken into account by calibrating using the same mode and gain setup as in the intended use.***

Sample programs are available to illustrate how to read and calibrate the various A/D inputs for the single-ended operating mode.

Mode	Read	Calibrate
Single-Ended, one channel	—	**AD_CAL_CHAN.C**
Single-Ended, all channels	**AD_RDVOLT_ALL.C**	**AD_CAL_ALL.C**

B.4.4 Serial Communication

The Prototyping Board allows you to access the serial ports from the RCM4300 module. Table B-5 summarizes the configuration options. Note that Serial Ports F can be used only when the RCM4310 is installed on the Prototyping Board.

Table B-5. Prototyping Board Serial Port Configurations

Serial Port	Header	Default Use	Alternate Use
A	J2	Programming Port	RS-232
B	J2	Serial Flash, *microSD™ Card*, A/D Converter (if equipped)	RS-232
C	J2, J4	RS-232	—
D	J2, J4	RS-232	—
E	J2	RS-232	—
F	J2	RS-232	—

Serial Ports E and F may be used as serial ports, or the corresponding pins at header location J2 may be used as parallel ports.

B.4.4.1 RS-232

RS-232 serial communication on header J4 on both Prototyping Boards is supported by an RS-232 transceiver installed at U3. This transceiver provides the voltage output, slew rate, and input voltage immunity required to meet the RS-232 serial communication protocol. Basically, the chip translates the Rabbit 4000's signals to RS-232 signal levels. Note that the polarity is reversed in an RS-232 circuit so that a +3.3 V output becomes approximately -10 V and 0 V is output as +10 V. The RS-232 transceiver also provides the proper line loading for reliable communication.

RS-232 can be used effectively at the RCM4300 module's maximum baud rate for distances of up to 15 m.

RS-232 flow control on an RS-232 port is initiated in software using the **serXflowcontrolOn** function call from **RS232.LIB**, where **X** is the serial port (C or D). The locations of the flow control lines are specified using a set of five macros.

SERX_RTS_PORT—Data register for the parallel port that the RTS line is on (e.g., PCDR).

SERA_RTS_SHADOW—Shadow register for the RTS line's parallel port (e.g., PCDRShadow).

SERA_RTS_BIT—The bit number for the RTS line.

SERA_CTS_PORT—Data register for the parallel port that the CTS line is on (e.g., PCDRShadow).

SERA_CTS_BIT—The bit number for the CTS line.

Standard 3-wire RS-232 communication using Serial Ports C and D is illustrated in the following sample code.

```
#define CINBUFSIZE  15   // set size of circular buffers in bytes
#define COUTBUFSIZE 15

#define DINBUFSIZE  15
#define DOUTBUFSIZE 15

#define MYBAUD 115200   // set baud rate
#endif

main(){
    serCopen(_MYBAUD);   // open Serial Ports C and D
    serDopen(_MYBAUD);
    serCwrFlush();       // flush their input and transmit buffers
    serCrdFlush();
    serDwrFlush();
    serDrdFlush();
    serCclose(_MYBAUD);  // close Serial Ports C and D
    serDclose(_MYBAUD);
}
```

B.5 Prototyping Board Jumper Configurations

Figure B-8 shows the header locations used to configure the various Prototyping Board options via jumpers.

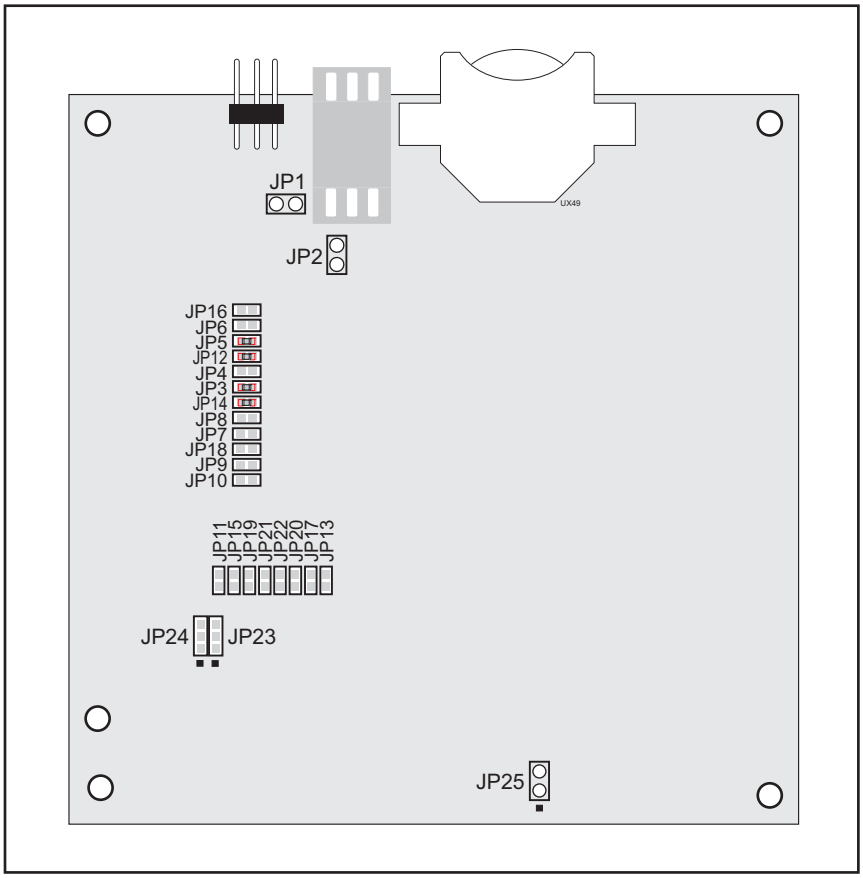


Figure B-8. Location of Configurable Jumpers on Prototyping Board

Table B-6 lists the configuration options using either jumpers or 0 Ω surface-mount resistors.

Table B-6. RCM4300 Prototyping Board Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	+5 V Current Measurement	1–2	Via trace or jumper	Connected
JP2	+3.3 V Current Measurement	1–2	Via trace or jumper	Connected
JP3 JP4	PC0/TxD/LED DS2	JP3 1–2	TxD on header J4	✕
		JP4 1–2	PC0 to LED DS2	
		n.c.	PC0 available on header J2	

Table B-6. RCM4300 Prototyping Board Jumper Configurations (continued)

Header	Description	Pins Connected		Factory Default
JP5 JP6	PC1/RxD/Switch S2	JP5 1–2	RxD on header J4	✕
		JP6 1–2	PC1 to Switch S2	
		n.c.	PC1 available on header J2	
JP7 JP8	PC2/TxC/LED DS3	JP7 1–2	TxC on header J4	✕
		JP6 1–2	PC2 to LED DS3	
		n.c.	PC2 available on header J2	
JP9 JP10	PC3/RxC/Switch S3	JP9 1–2	PC3 to Switch S3	
		JP10 1–2	RxC on header J4	✕
		n.c.	PC3 available on header J2	
JP11	LN0 buffer/filter to RCM4300	1–2		Connected
JP12	PB2/LED DS2	1–2	Connected: PB2 to LED DS2	✕
		n.c.	PB2 available on header J2	
JP13	LN1 buffer/filter to RCM4300	1–2		Connected
JP14	PB3/LED DS3	1–2	Connected: PB3 to LED DS3	✕
		n.c.	PB3 available on header J2	
JP15	LN2 buffer/filter to RCM4300	1–2		Connected
JP16	PB4/Switch S2	1–2	Connected: PB4 to Switch S2	✕
		n.c.	PB4 available on header J2	
JP17	LN3 buffer/filter to RCM4300	1–2		Connected
JP18	PB5/Switch S3	1–2	Connected: PB5 to Switch S3	✕
		n.c.	PB5 available on header J2	
JP19	LN4 buffer/filter to RCM4300	1–2		Connected
JP20	LN5 buffer/filter to RCM4300	1–2		Connected
JP21	LN6 buffer/filter to RCM4300	1–2		Connected
JP22	LN7 buffer/filter to RCM4300	1–2		Connected

Table B-6. RCM4300 Prototyping Board Jumper Configurations (continued)

Header	Description	Pins Connected		Factory Default
JP23	LN4_IN–LN6_IN	1–2	Tied to analog ground	✕
		2–3	Tied to VREF	
JP24	LN0_IN–LN3_IN	1–2	Tied to analog ground	✕
		2–3	Tied to VREF	
JP25	Thermistor Location	1–2		n.c.

NOTE: Jumper connections JP3–JP10, JP12, JP14, JP16, JP18, JP23, and JP24 are made using 0 Ω surface-mounted resistors. Jumper connections JP11, JP13, JP15, JP17, and JP19–JP22 are made using 470 Ω surface-mounted resistors.

APPENDIX C. POWER SUPPLY

Appendix C provides information on the current requirements of the RCM4300, and includes some background on the chip select circuit used in power management.

C.1 Power Supplies

The RCM4300 requires a regulated 3.0 V – 3.6 V DC power source. The RabbitCore design presumes that the voltage regulator is on the user board, and that the power is made available to the RCM4300 board through header J2.

An RCM4300 with no loading at the outputs operating at 58.98 MHz typically draws 240 mA, and may draw up to 275 mA at 3.6 V and 85°C; the corresponding current draw for the RCM4210 is typically 200 mA, and up to 225 mA at 3.6 V and 85°C.

C.1.1 Battery Backup

The RCM4300 does not have a battery, but there is provision for a customer-supplied battery to back up the data SRAM and keep the internal Rabbit 4000 real-time clock running.

Header J2, shown in Figure C-1, allows access to the external battery. This header makes it possible to connect an external 3 V power supply. This allows the SRAM and the internal Rabbit 4000 real-time clock to retain data with the RCM4300 powered down.

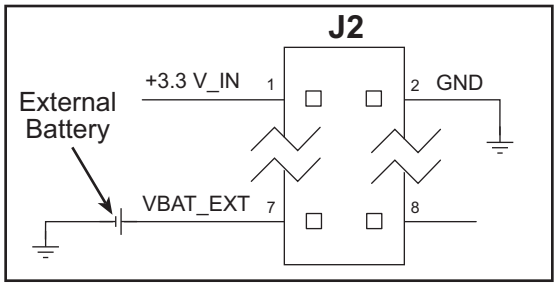


Figure C-1. External Battery Connections at Header J2

A lithium battery with a nominal voltage of 3 V and a minimum capacity of 165 mA·h is recommended. A lithium battery is strongly recommended because of its nearly constant nominal voltage over most of its life.

The drain on the battery by the RCM4300 is typically 7.5 µA when no other power is supplied. If a 165 mA·h battery is used, the battery can last about 2.5 years:

$$\frac{165\ \text{mA}\cdot\text{h}}{7.5\ \mu\text{A}} = 2.5\ \text{years}.$$

The actual battery life in your application will depend on the current drawn by components not on the RCM4300 and on the storage capacity of the battery. The RCM4300 does not drain the battery while it is powered up normally.

Cycle the main power off/on after you install a backup battery for the first time, and whenever you replace the battery. This step will minimize the current drawn by the real-time clock oscillator circuit from the backup battery should the RCM4300 experience a loss of main power.

NOTE: Remember to cycle the main power off/on any time the RCM4300 is removed from the Prototyping Board or motherboard since that is where the backup battery would be located.

Rabbit Semiconductor's Technical Note TN235, ***External 32.768 kHz Oscillator Circuits***, provides additional information about the current draw by the real-time clock oscillator circuit.

C.1.2 Battery-Backup Circuit

Figure C-2 shows the battery-backup circuit.

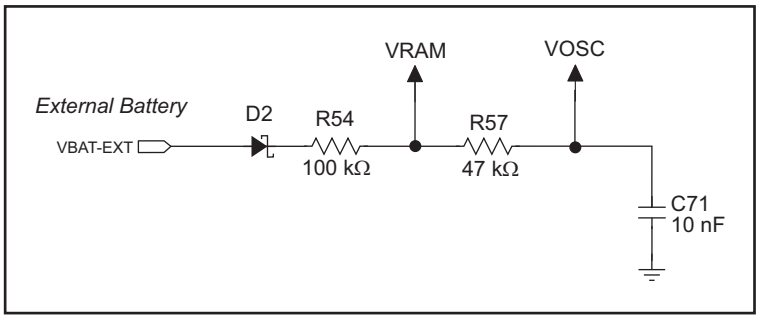


Figure C-2. RCM4300 Backup Battery Circuit

The battery-backup circuit serves three purposes:

- It reduces the battery voltage to the SRAM and to the real-time clock, thereby limiting the current consumed by the real-time clock and lengthening the battery life.
- It ensures that current can flow only ***out*** of the battery to prevent charging the battery.
- A voltage, VOSC, is supplied to U14, which keeps the 32.768 kHz oscillator working when the voltage begins to drop.

C.1.3 Reset Generator

The RCM4300 uses a reset generator to reset the Rabbit 4000 microprocessor when the voltage drops below the voltage necessary for reliable operation. The reset occurs between 2.85 V and 3.00 V, typically 2.93 V. Since the RCM4300 will operate at voltages as low as 3.0 V, exercise care when operating close to the 3.0 V minimum voltage (for example, keep the power supply as close as possible to the RCM4300) since your RCM4300 could reset unintentionally.

The RCM4300 has a reset output, pin 3 on header J2.

INDEX

A

A/D converter
  access via Prototyping Board 110
  function calls
    anaIn 60
    anaInCalib 62
    anaInConfig 56
    anaInDiff 66
    anaInDriver 58
    anaInEERd 69
    anaInEEWr 71
    anaInmAmps 68
    anaInVolts 64
  inputs
    differential measurements . 111
    negative voltages 111
    single-ended measurements 110
additional information
  online documentation 10
analog inputs *See* A/D converter
auxiliary I/O bus 35

B

battery backup
  battery life 119
  circuit 119
  external battery connections . 118
  real-time clock 119
  reset generator 120
  use of battery-backed SRAM 52
board initialization
  function calls 54
    brdInit 54
bus loading 95

C

clock doubler 44
cloning 53
compiler options 16, 47
conformal coating 99, 100

D

Development Kit 9
  512 MB microSD™ Card ... 9
  AC adapter 9
  DC power supply 9
  Getting Started instructions 9
  programming cable 9
digital I/O 29
  function calls 49
    digInAlert 55
    timedAlert 55
  I/O buffer sourcing and sinking limits 95
  memory interface 35
  SMODE0 ..... 35, 38
  SMODE1 ..... 35, 38
dimensions
  Prototyping Board 105
  RCM4300 88
Dynamic C 10, 11, 16, 47
  add-on modules 11, 73
    installation 11
  battery-backed SRAM 52
  compiler options 16, 47
  FAT file system 53
  libraries
    BOOTDEV_SFLASH.LIB 50
    PACKET.LIB 49
    RCM43xx.LIB 54
    RS232.LIB 49
    SDFLASH.LIB 53
  protected variables 52
  sample programs 19
  standard features
    debugging 48
  telephone-based technical support 10, 73
  upgrades and patches 73

E

Ethernet cables 74
  how to tell them apart 74
Ethernet connections 74, 76
  10/100Base-T 76
  10Base-T Ethernet card 74
  additional resources 86
  direct connection 76
  Ethernet cables 76
  Ethernet hub 74
  IP addresses 76, 78
  MAC addresses 79
  steps 75
Ethernet port 37
  pinout 37
exclusion zone 89

F

features 7
  Prototyping Boards . 102, 103

H

hardware connections
  install RCM4300 on Prototyping Board 13
  power supply 15
  programming cable 14

I

I/O buffer sourcing and sinking limits 95
IP addresses 78
  how to set in sample programs 83
  how to set PC IP address .. 84

J

jumper configurations
  Prototyping Board 115
    JP1 (+5 V current measurement) 115

JP1 (LN0 buffer/filter to RCM4300) 116
JP12 (PB2/LED DS2) . 116
JP13 (LN1 buffer/filter to RCM4300) 116
JP14 (PB3/LED DS3) . 116
JP15 (LN2 buffer/filter to RCM4300) 116
JP16 (PB4/Switch S2) . 116
JP17 (LN3 buffer/filter to RCM4300) 116
JP18 (PB5/Switch S2) . 116
JP19 (LN4 buffer/filter to RCM4300) 116
JP2 (+ 3.3 V current measurement) 115
JP20 (LN5 buffer/filter to RCM4300) 116
JP21 (LN6 buffer/filter to RCM4300) 116
JP22 (LN7 buffer/filter to RCM4300) 116
JP23 (analog inputs LN4–LN6 configuration) ... 117
JP24 (analog inputs LN0–LN3 configuration) ... 117
JP3–JP4 (PC0/TxD/LED DS2) 115
JP5–JP6 (PC1/RxD/Switch S2) 116
JP7–JP8 (PC2/TxC/LED DS3) 116
JP9–JP10 (PC3/RxC/Switch S3) 116
RCM4300 98
JP1 (PE6 or SMODE1 output on J4) 98
JP10 (LN5 or PD5 on J4) .. 99
JP11 (LN7 or PD7 on J4) .. 99
JP2 (PE5 or SMODE0 output on J4) 98
JP3 (PE7 or STATUS output on J4) 98
JP4 (LN0 or PD0 on J4) 98
JP5 (LN1 or PD1 on J4) 98
JP6 (LN2 or PD2 on J4) 99
JP7 (LN3 or PD3 on J4) 99
JP8 (LN4 or PD4 on J4) 99
JP9 (LN6 or PD6 on J4) 99
jumper locations 98

M

MAC addresses 79
microSD™ Card
adapter 46
formatting 53
software
SDFLASH.LIB 53
use in mass-storage application 53
use with SD Card reader ... 46

P

pinout
Ethernet port 37
Prototyping Board 107
RCM4300 29
alternate configurations . 31
power supplies
+3.3 V 118
battery backup 118
Program Mode 39
switching modes 39
programming cable
PROG connector 39
RCM4300 connections 14
programming port 38
Prototyping Board 102
access to RCM4300 analog inputs 103
adding components 109
dimensions 105
expansion area 103
features 102, 103
jumper configurations 115
jumper locations 115
mounting RCM4300 13
pinout 107
power supply 106
prototyping area 108
specifications 106
use of Rabbit 4000 signals 108

R

Rabbit 4000
spectrum spreader time delays 97
tamper detection 20, 45
VBAT RAM memory . 20, 45
Rabbit subsystems 30
RCM4300
mounting on Prototyping Board 13
real-time clock
battery backup 119
Run Mode 39
switching modes 39

S

sample programs 19
A/D converter
AD_CAL_ALL.C . 24, 112
AD_CAL_CHAN.C 24, 112
AD_RDVOLT_ALL.C 24, 112
AD_SAMPLE.C 24
THERMISTOR.C . 25, 112
A/D converter calibration
DNLOADCALIB.C 25
UPLOADCALIB.C 25
FAT file system 27
getting to know the RCM4300
CONTROLLED.C 19
FLASHLED1.C 19
FLASHLED2.C 19
TOGGLESWITCH.C 20
how to run TCP/IP sample programs 82, 83
how to set IP address 83
microSD™ Card
SDFLASH_INSPECT.c 20
PONG.C 16
real-time clock
RTC_TEST.C 27
SETRTCKB.C 27
serial communication
FLOWCONTROL.C 21
IOCONFIG_SWITCHECHO.C 23
PARITY.C 21
SERDMA.C 21
SIMPLE3WIRE.C 22
SIMPLE5WIRE.C 22
SWITCHCHAR.C 22
TCP/IP
BROWSELED.C 85
DISPLAY_MAC.C 79
PINGLED.C 85
PINGME.C 85
SMTP.C 85
USERBLOCK_CLEAR.C 52
USERBLOCK_INFO.C 52
SD Card adapter 46
serial communication 36
function calls 49
Prototyping Board

RS-232 114
software
PACKET.LIB 49
RS232.LIB 49
serial flash memory
function calls 50
sbfRead 50
sbfWriteFlash 51
software
BOOTDEV_SFLASH.LIB 50
serial ports 36
Ethernet port 37
programming port 38
Serial Port B (shared) 36
Serial Ports E/F
configuration information . 23, 36
software 10
auxiliary I/O bus 35, 49
I/O drivers 49
libraries
ADC_ADS7870.LIB 56
BOOTDEV_SFLASH.LIB 50
RCM43xx.LIB 54
SDFLASH.LIB 53
microSD™ Card 53
serial communication drivers 49
serial flash boot drivers 50
specifications 87
A/D converter chip 92
bus loading 95
digital I/O buffer sourcing and sinking limits 95
dimensions 88
electrical, mechanical, and environmental 90
exclusion zone 89
header footprint 93
headers 93
Prototyping Board 106
Rabbit 4000 DC characteristics 94
Rabbit 4000 timing diagram . 96
relative pin 1 locations 93
spectrum spreader 97
settings 44
subsystems
digital inputs and outputs .. 29
switching modes 39

T

tamper detection 20, 45
TCP/IP primer 76
technical support 17

U

user block
determining size 52
function calls 52
readUserBlock 45
writeUserBlock 45
reserved area for calibration constants 52

V

VBAT RAM memory 20, 45

SCHEMATICS

090-0229 RCM4300 Schematic

www.rabbit.com/documentation/schemat/090-0229.pdf

090-0230 Prototyping Board Schematic

www.rabbit.com/documentation/schemat/090-0230.pdf

090-0252 USB Programming Cable Schematic

www.rabbit.com/documentation/schemat/090-0252.pdf

You may use the URL information provided above to access the latest schematics directly.

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Rabbit Semiconductor:

20-101-1153